

libstdc++

Generated by Doxygen 1.8.9.1

Tue Jun 30 2015 12:19:40

## Contents

<b>1</b>	<b>Todo List</b>	<b>1</b>
<b>2</b>	<b>Module Documentation</b>	<b>4</b>
2.1	Adaptors for pointers to functions . . . . .	4
2.1.1	Detailed Description . . . . .	5
2.1.2	Function Documentation . . . . .	5
2.2	Adaptors for pointers to members . . . . .	6
2.2.1	Detailed Description . . . . .	6
2.3	Algorithms . . . . .	7
2.3.1	Detailed Description . . . . .	7
2.4	Allocators . . . . .	8
2.4.1	Detailed Description . . . . .	8
2.4.2	Typedef Documentation . . . . .	8
2.5	Arithmetic Classes . . . . .	10
2.5.1	Detailed Description . . . . .	10
2.6	Associative . . . . .	11
2.6.1	Detailed Description . . . . .	11
2.7	Atomics . . . . .	12
2.7.1	Detailed Description . . . . .	13
2.7.2	Macro Definition Documentation . . . . .	13
2.7.3	Typedef Documentation . . . . .	14
2.7.4	Enumeration Type Documentation . . . . .	17
2.7.5	Function Documentation . . . . .	17
2.8	Base and Implementation Classes . . . . .	18
2.8.1	Detailed Description . . . . .	20
2.9	Base and Implementation Classes . . . . .	21
2.9.1	Detailed Description . . . . .	22
2.9.2	Typedef Documentation . . . . .	22
2.9.3	Enumeration Type Documentation . . . . .	23
2.9.4	Function Documentation . . . . .	23
2.9.5	Variable Documentation . . . . .	23
2.10	Base and Policy Classes . . . . .	24
2.10.1	Detailed Description . . . . .	24
2.10.2	Enumeration Type Documentation . . . . .	24
2.11	Base and Policy Classes . . . . .	25
2.11.1	Detailed Description . . . . .	25

2.12 Base and Policy Classes . . . . .	26
2.12.1 Detailed Description . . . . .	26
2.13 Bernoulli Distributions . . . . .	27
2.13.1 Detailed Description . . . . .	28
2.13.2 Function Documentation . . . . .	28
2.14 Binary Search . . . . .	30
2.14.1 Detailed Description . . . . .	30
2.14.2 Function Documentation . . . . .	31
2.15 Binder Classes . . . . .	35
2.15.1 Detailed Description . . . . .	35
2.15.2 Function Documentation . . . . .	35
2.16 Boolean Operations Classes . . . . .	37
2.16.1 Detailed Description . . . . .	37
2.17 Branch-Based . . . . .	38
2.17.1 Detailed Description . . . . .	38
2.18 Comparison Classes . . . . .	39
2.18.1 Detailed Description . . . . .	39
2.19 Concurrency . . . . .	40
2.20 Containers . . . . .	41
2.20.1 Detailed Description . . . . .	41
2.21 Containers . . . . .	42
2.21.1 Detailed Description . . . . .	42
2.22 Data Structure Type . . . . .	43
2.22.1 Detailed Description . . . . .	43
2.23 Diagnostics . . . . .	44
2.24 Exceptions . . . . .	45
2.24.1 Detailed Description . . . . .	45
2.25 Exceptions . . . . .	46
2.25.1 Detailed Description . . . . .	46
2.25.2 Function Documentation . . . . .	46
2.26 Extensions . . . . .	48
2.26.1 Detailed Description . . . . .	48
2.27 Function Objects . . . . .	49
2.27.1 Detailed Description . . . . .	50
2.28 Hash-Based . . . . .	51
2.28.1 Detailed Description . . . . .	51
2.29 Hashes . . . . .	52

2.29.1 Detailed Description . . . . .	52
2.30 Heap . . . . .	53
2.30.1 Detailed Description . . . . .	53
2.30.2 Function Documentation . . . . .	53
2.31 Heap-Based . . . . .	59
2.31.1 Detailed Description . . . . .	60
2.31.2 Function Documentation . . . . .	60
2.32 Invalidation Guarantees . . . . .	61
2.32.1 Detailed Description . . . . .	61
2.33 Iterator Tags . . . . .	62
2.33.1 Detailed Description . . . . .	62
2.34 Iterators . . . . .	63
2.34.1 Detailed Description . . . . .	65
2.34.2 Function Documentation . . . . .	65
2.35 List-Based . . . . .	68
2.35.1 Detailed Description . . . . .	68
2.36 Locales . . . . .	69
2.36.1 Detailed Description . . . . .	69
2.37 Mutating . . . . .	70
2.37.1 Detailed Description . . . . .	72
2.37.2 Function Documentation . . . . .	72
2.38 Negators . . . . .	93
2.38.1 Detailed Description . . . . .	93
2.38.2 Function Documentation . . . . .	93
2.39 Non-Mutating . . . . .	95
2.39.1 Detailed Description . . . . .	96
2.39.2 Function Documentation . . . . .	96
2.40 Normal Distributions . . . . .	108
2.40.1 Detailed Description . . . . .	109
2.40.2 Function Documentation . . . . .	109
2.41 Numeric_arrays . . . . .	111
2.41.1 Detailed Description . . . . .	114
2.41.2 Function Documentation . . . . .	114
2.42 Pointer_abstractions . . . . .	123
2.42.1 Detailed Description . . . . .	125
2.42.2 Function Documentation . . . . .	125
2.43 Poisson Distributions . . . . .	127



2.43.1 Detailed Description . . . . .	128
2.43.2 Function Documentation . . . . .	128
2.44 Policy-Based Data Structures . . . . .	131
2.44.1 Detailed Description . . . . .	131
2.45 Random Number Distributions . . . . .	132
2.45.1 Detailed Description . . . . .	132
2.46 Random Number Generation . . . . .	133
2.46.1 Detailed Description . . . . .	133
2.46.2 Function Documentation . . . . .	133
2.47 Random Number Generators . . . . .	134
2.47.1 Detailed Description . . . . .	135
2.47.2 Typedef Documentation . . . . .	135
2.47.3 Function Documentation . . . . .	136
2.48 Random Number Utilities . . . . .	139
2.48.1 Detailed Description . . . . .	139
2.49 Regular Expressions . . . . .	140
2.49.1 Detailed Description . . . . .	145
2.49.2 Typedef Documentation . . . . .	145
2.49.3 Enumeration Type Documentation . . . . .	146
2.49.4 Function Documentation . . . . .	146
2.49.5 Variable Documentation . . . . .	175
2.50 SGI . . . . .	179
2.51 Sequences . . . . .	180
2.51.1 Detailed Description . . . . .	180
2.52 Set Operation . . . . .	181
2.52.1 Detailed Description . . . . .	181
2.52.2 Function Documentation . . . . .	182
2.53 Sorting . . . . .	189
2.53.1 Detailed Description . . . . .	191
2.53.2 Function Documentation . . . . .	191
2.54 Strings . . . . .	210
2.54.1 Detailed Description . . . . .	210
2.54.2 Typedef Documentation . . . . .	210
2.55 Tags . . . . .	211
2.55.1 Detailed Description . . . . .	211
2.55.2 Typedef Documentation . . . . .	211
2.56 Traits . . . . .	212

2.56.1	Detailed Description	213
2.56.2	Enumeration Type Documentation	213
2.57	Uniform Distributions	214
2.57.1	Detailed Description	214
2.57.2	Function Documentation	214
2.58	Unordered Associative	217
2.58.1	Detailed Description	217
2.59	Utilities	218
2.59.1	Detailed Description	219
2.59.2	Function Documentation	219
2.59.3	Variable Documentation	222
<b>3</b>	<b>Namespace Documentation</b>	<b>223</b>
3.1	__gnu_cxx Namespace Reference	223
3.1.1	Detailed Description	229
3.1.2	Function Documentation	229
3.2	__gnu_cxx::__detail Namespace Reference	238
3.2.1	Detailed Description	239
3.2.2	Function Documentation	239
3.3	__gnu_cxx::typelist Namespace Reference	239
3.3.1	Detailed Description	240
3.3.2	Function Documentation	240
3.4	__gnu_debug Namespace Reference	240
3.4.1	Detailed Description	243
3.4.2	Enumeration Type Documentation	243
3.4.3	Function Documentation	243
3.5	__gnu_internal Namespace Reference	245
3.5.1	Detailed Description	245
3.6	__gnu_parallel Namespace Reference	246
3.6.1	Detailed Description	253
3.6.2	Typedef Documentation	253
3.6.3	Enumeration Type Documentation	253
3.6.4	Function Documentation	254
3.6.5	Variable Documentation	292
3.7	__gnu_pbds Namespace Reference	292
3.7.1	Detailed Description	294
3.8	__gnu_profile Namespace Reference	295

3.8.1	Detailed Description	298
3.8.2	Typedef Documentation	298
3.8.3	Function Documentation	298
3.9	<code>__gnu_sequential</code> Namespace Reference	299
3.9.1	Detailed Description	299
3.10	<code>abi</code> Namespace Reference	299
3.10.1	Detailed Description	299
3.11	<code>std</code> Namespace Reference	299
3.11.1	Detailed Description	353
3.11.2	Typedef Documentation	353
3.11.3	Enumeration Type Documentation	354
3.11.4	Function Documentation	354
3.12	<code>std::__debug</code> Namespace Reference	409
3.12.1	Detailed Description	411
3.13	<code>std::__detail</code> Namespace Reference	411
3.13.1	Detailed Description	413
3.14	<code>std::__parallel</code> Namespace Reference	413
3.14.1	Detailed Description	427
3.15	<code>std::__profile</code> Namespace Reference	427
3.15.1	Detailed Description	429
3.16	<code>std::regex_constants</code> Namespace Reference	429
3.16.1	Detailed Description	430
3.17	<code>std::rel_ops</code> Namespace Reference	430
3.17.1	Detailed Description	430
3.17.2	Function Documentation	430
3.18	<code>std::tr1</code> Namespace Reference	432
3.18.1	Detailed Description	432
3.19	<code>std::tr1::__detail</code> Namespace Reference	432
3.19.1	Detailed Description	432
3.20	<code>std::tr2</code> Namespace Reference	432
3.20.1	Detailed Description	432
3.21	<code>std::tr2::__detail</code> Namespace Reference	432
3.21.1	Detailed Description	432
<b>4</b>	<b>Class Documentation</b>	<b>432</b>
4.1	<code>__cxxabiv1::__forced_unwind</code> Class Reference	432
4.1.1	Detailed Description	432

4.2	<a href="#">__gnu_cxx::__alloc_traits&lt;_Alloc&gt; Struct Template Reference</a>	433
4.2.1	Detailed Description	434
4.2.2	Member Typedef Documentation	434
4.2.3	Member Function Documentation	435
4.3	<a href="#">__gnu_cxx::__common_pool_policy&lt;_PoolTp, _Thread&gt; Struct Template Reference</a>	438
4.3.1	Detailed Description	438
4.4	<a href="#">__gnu_cxx::__detail::__mini_vector&lt;_Tp&gt; Class Template Reference</a>	438
4.4.1	Detailed Description	438
4.5	<a href="#">__gnu_cxx::__detail::__Bitmap_counter&lt;_Tp&gt; Class Template Reference</a>	439
4.5.1	Detailed Description	439
4.6	<a href="#">__gnu_cxx::__detail::__Ffit_finder&lt;_Tp&gt; Class Template Reference</a>	440
4.6.1	Detailed Description	440
4.6.2	Member Typedef Documentation	440
4.7	<a href="#">__gnu_cxx::__mt_alloc&lt;_Tp, _Poolp&gt; Class Template Reference</a>	441
4.7.1	Detailed Description	442
4.8	<a href="#">__gnu_cxx::__mt_alloc_base&lt;_Tp&gt; Class Template Reference</a>	442
4.8.1	Detailed Description	443
4.9	<a href="#">__gnu_cxx::__per_type_pool_policy&lt;_Tp, _PoolTp, _Thread&gt; Struct Template Reference</a>	443
4.9.1	Detailed Description	443
4.10	<a href="#">__gnu_cxx::__pool&lt;_Thread&gt; Class Template Reference</a>	443
4.10.1	Detailed Description	443
4.11	<a href="#">__gnu_cxx::__pool&lt;false&gt; Class Template Reference</a>	444
4.11.1	Detailed Description	444
4.12	<a href="#">__gnu_cxx::__pool&lt;true&gt; Class Template Reference</a>	445
4.12.1	Detailed Description	446
4.13	<a href="#">__gnu_cxx::__pool_alloc&lt;_Tp&gt; Class Template Reference</a>	446
4.13.1	Detailed Description	447
4.14	<a href="#">__gnu_cxx::__pool_alloc_base Class Reference</a>	447
4.14.1	Detailed Description	448
4.15	<a href="#">__gnu_cxx::__pool_base Struct Reference</a>	449
4.15.1	Detailed Description	449
4.16	<a href="#">__gnu_cxx::__rc_string_base&lt;_CharT, _Traits, _Alloc&gt; Class Template Reference</a>	449
4.16.1	Detailed Description	451
4.17	<a href="#">__gnu_cxx::__scoped_lock Class Reference</a>	451
4.17.1	Detailed Description	452
4.18	<a href="#">__gnu_cxx::__versa_string&lt;_CharT, _Traits, _Alloc, _Base&gt; Class Template Reference</a>	452
4.18.1	Detailed Description	455

4.18.2	Constructor & Destructor Documentation	455
4.18.3	Member Function Documentation	458
4.18.4	Member Data Documentation	508
4.19	<code>__gnu_cxx::_Caster&lt;_ToType&gt;</code> Struct Template Reference	508
4.19.1	Detailed Description	508
4.20	<code>__gnu_cxx::_Char_types&lt;_CharT&gt;</code> Struct Template Reference	509
4.20.1	Detailed Description	509
4.21	<code>__gnu_cxx::_ExtPtr_allocator&lt;_Tp&gt;</code> Class Template Reference	509
4.21.1	Detailed Description	510
4.22	<code>__gnu_cxx::_Invalid_type</code> Struct Reference	510
4.22.1	Detailed Description	510
4.23	<code>__gnu_cxx::_Pointer_adapter&lt;_Storage_policy&gt;</code> Class Template Reference	510
4.23.1	Detailed Description	512
4.24	<code>__gnu_cxx::_Relative_pointer_impl&lt;_Tp&gt;</code> Class Template Reference	512
4.24.1	Detailed Description	513
4.25	<code>__gnu_cxx::_Relative_pointer_impl&lt;const _Tp&gt;</code> Class Template Reference	513
4.25.1	Detailed Description	513
4.26	<code>__gnu_cxx::_Std_pointer_impl&lt;_Tp&gt;</code> Class Template Reference	514
4.26.1	Detailed Description	514
4.27	<code>__gnu_cxx::_Unqualified_type&lt;_Tp&gt;</code> Struct Template Reference	514
4.27.1	Detailed Description	514
4.28	<code>__gnu_cxx::annotate_base</code> Struct Reference	515
4.28.1	Detailed Description	515
4.29	<code>__gnu_cxx::array_allocator&lt;_Tp, _Array&gt;</code> Class Template Reference	516
4.29.1	Detailed Description	516
4.30	<code>__gnu_cxx::array_allocator_base&lt;_Tp&gt;</code> Class Template Reference	517
4.30.1	Detailed Description	517
4.31	<code>__gnu_cxx::bitmap_allocator&lt;_Tp&gt;</code> Class Template Reference	518
4.31.1	Detailed Description	519
4.31.2	Member Function Documentation	519
4.32	<code>__gnu_cxx::char_traits&lt;_CharT&gt;</code> Struct Template Reference	520
4.32.1	Detailed Description	521
4.33	<code>__gnu_cxx::character&lt;V, I, S&gt;</code> Struct Template Reference	521
4.33.1	Detailed Description	521
4.34	<code>__gnu_cxx::condition_base</code> Struct Reference	522
4.34.1	Detailed Description	522
4.35	<code>__gnu_cxx::debug_allocator&lt;_Alloc&gt;</code> Class Template Reference	522

4.35.1 Detailed Description . . . . .	522
4.36 <a href="#">__gnu_cxx::enc_filebuf&lt;_CharT&gt; Class Template Reference</a> . . . . .	523
4.36.1 Detailed Description . . . . .	523
4.37 <a href="#">__gnu_cxx::encoding_char_traits&lt;_CharT&gt; Struct Template Reference</a> . . . . .	524
4.37.1 Detailed Description . . . . .	525
4.38 <a href="#">__gnu_cxx::encoding_state Class Reference</a> . . . . .	525
4.38.1 Detailed Description . . . . .	526
4.39 <a href="#">__gnu_cxx::forced_error Struct Reference</a> . . . . .	526
4.39.1 Detailed Description . . . . .	526
4.40 <a href="#">__gnu_cxx::free_list Class Reference</a> . . . . .	526
4.40.1 Detailed Description . . . . .	527
4.40.2 Member Function Documentation . . . . .	527
4.41 <a href="#">__gnu_cxx::limit_condition Struct Reference</a> . . . . .	528
4.41.1 Detailed Description . . . . .	528
4.42 <a href="#">__gnu_cxx::limit_condition::always_adjustor Struct Reference</a> . . . . .	528
4.42.1 Detailed Description . . . . .	528
4.43 <a href="#">__gnu_cxx::limit_condition::limit_adjustor Struct Reference</a> . . . . .	529
4.43.1 Detailed Description . . . . .	529
4.44 <a href="#">__gnu_cxx::limit_condition::never_adjustor Struct Reference</a> . . . . .	529
4.44.1 Detailed Description . . . . .	529
4.45 <a href="#">__gnu_cxx::malloc_allocator&lt;_Tp&gt; Class Template Reference</a> . . . . .	529
4.45.1 Detailed Description . . . . .	530
4.46 <a href="#">__gnu_cxx::new_allocator&lt;_Tp&gt; Class Template Reference</a> . . . . .	530
4.46.1 Detailed Description . . . . .	531
4.47 <a href="#">__gnu_cxx::random_condition Struct Reference</a> . . . . .	531
4.47.1 Detailed Description . . . . .	532
4.48 <a href="#">__gnu_cxx::random_condition::always_adjustor Struct Reference</a> . . . . .	532
4.48.1 Detailed Description . . . . .	532
4.49 <a href="#">__gnu_cxx::random_condition::group_adjustor Struct Reference</a> . . . . .	532
4.49.1 Detailed Description . . . . .	533
4.50 <a href="#">__gnu_cxx::random_condition::never_adjustor Struct Reference</a> . . . . .	533
4.50.1 Detailed Description . . . . .	533
4.51 <a href="#">__gnu_cxx::recursive_init_error Class Reference</a> . . . . .	533
4.51.1 Detailed Description . . . . .	533
4.52 <a href="#">__gnu_cxx::stdio_filebuf&lt;_CharT, _Traits&gt; Class Template Reference</a> . . . . .	533
4.52.1 Detailed Description . . . . .	534
4.52.2 Constructor & Destructor Documentation . . . . .	534

4.52.3	Member Function Documentation	535
4.53	<code>__gnu_cxx::stdio_sync_filebuf&lt;_CharT, _Traits&gt;</code> Class Template Reference	535
4.53.1	Detailed Description	536
4.53.2	Member Function Documentation	537
4.54	<code>__gnu_cxx::throw_allocator_base&lt;_Tp, _Cond&gt;</code> Class Template Reference	537
4.54.1	Detailed Description	538
4.55	<code>__gnu_cxx::throw_allocator_limit&lt;_Tp&gt;</code> Struct Template Reference	539
4.55.1	Detailed Description	540
4.56	<code>__gnu_cxx::throw_allocator_random&lt;_Tp&gt;</code> Struct Template Reference	541
4.56.1	Detailed Description	542
4.57	<code>__gnu_cxx::throw_value_base&lt;_Cond&gt;</code> Struct Template Reference	542
4.57.1	Detailed Description	543
4.58	<code>__gnu_cxx::throw_value_limit</code> Struct Reference	543
4.58.1	Detailed Description	544
4.59	<code>__gnu_cxx::throw_value_random</code> Struct Reference	545
4.59.1	Detailed Description	546
4.60	<code>__gnu_debug::_After_nth_from&lt;_Iterator&gt;</code> Class Template Reference	546
4.60.1	Detailed Description	546
4.61	<code>__gnu_debug::_BeforeBeginHelper&lt;_Sequence&gt;</code> Struct Template Reference	546
4.61.1	Detailed Description	546
4.62	<code>__gnu_debug::_Equal_to&lt;_Type&gt;</code> Class Template Reference	547
4.62.1	Detailed Description	547
4.63	<code>__gnu_debug::_Not_equal_to&lt;_Type&gt;</code> Class Template Reference	547
4.63.1	Detailed Description	547
4.64	<code>__gnu_debug::_Safe_iterator&lt;_Iterator, _Sequence&gt;</code> Class Template Reference	548
4.64.1	Detailed Description	549
4.64.2	Constructor & Destructor Documentation	550
4.64.3	Member Function Documentation	551
4.64.4	Member Data Documentation	555
4.65	<code>__gnu_debug::_Safe_iterator_base</code> Class Reference	556
4.65.1	Detailed Description	557
4.65.2	Constructor & Destructor Documentation	557
4.65.3	Member Function Documentation	558
4.65.4	Member Data Documentation	559
4.66	<code>__gnu_debug::_Safe_local_iterator&lt;_Iterator, _Sequence&gt;</code> Class Template Reference	560
4.66.1	Detailed Description	561
4.66.2	Constructor & Destructor Documentation	562

4.66.3	Member Function Documentation	562
4.66.4	Member Data Documentation	566
4.67	<code>__gnu_debug::__Safe_local_iterator_base</code> Class Reference	567
4.67.1	Detailed Description	568
4.67.2	Constructor & Destructor Documentation	568
4.67.3	Member Function Documentation	569
4.67.4	Member Data Documentation	570
4.68	<code>__gnu_debug::__Safe_sequence&lt;_Sequence&gt;</code> Class Template Reference	571
4.68.1	Detailed Description	571
4.68.2	Member Function Documentation	572
4.68.3	Member Data Documentation	573
4.69	<code>__gnu_debug::__Safe_sequence_base</code> Class Reference	574
4.69.1	Detailed Description	575
4.69.2	Constructor & Destructor Documentation	575
4.69.3	Member Function Documentation	575
4.69.4	Member Data Documentation	576
4.70	<code>__gnu_debug::__Safe_unordered_container&lt;_Container&gt;</code> Class Template Reference	577
4.70.1	Detailed Description	578
4.70.2	Member Function Documentation	578
4.70.3	Member Data Documentation	580
4.71	<code>__gnu_debug::__Safe_unordered_container_base</code> Class Reference	581
4.71.1	Detailed Description	582
4.71.2	Constructor & Destructor Documentation	582
4.71.3	Member Function Documentation	582
4.71.4	Member Data Documentation	584
4.72	<code>__gnu_parallel::__accumulate_binop_reduct&lt;_BinOp&gt;</code> Struct Template Reference	584
4.72.1	Detailed Description	584
4.73	<code>__gnu_parallel::__accumulate_selector&lt;_It&gt;</code> Struct Template Reference	585
4.73.1	Detailed Description	585
4.73.2	Member Function Documentation	585
4.73.3	Member Data Documentation	586
4.74	<code>__gnu_parallel::__adjacent_difference_selector&lt;_It&gt;</code> Struct Template Reference	586
4.74.1	Detailed Description	586
4.74.2	Member Data Documentation	586
4.75	<code>__gnu_parallel::__adjacent_find_selector</code> Struct Reference	587
4.75.1	Detailed Description	587
4.75.2	Member Function Documentation	587



4.76	<a href="#">__gnu_parallel::__binder1st&lt; _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType &gt; Class Template Reference</a>	588
4.76.1	Detailed Description	589
4.76.2	Member Typedef Documentation	589
4.77	<a href="#">__gnu_parallel::__binder2nd&lt; _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType &gt; Class Template Reference</a>	590
4.77.1	Detailed Description	590
4.77.2	Member Typedef Documentation	591
4.78	<a href="#">__gnu_parallel::__count_if_selector&lt; _It, _Diff &gt; Struct Template Reference</a>	591
4.78.1	Detailed Description	591
4.78.2	Member Function Documentation	592
4.78.3	Member Data Documentation	592
4.79	<a href="#">__gnu_parallel::__count_selector&lt; _It, _Diff &gt; Struct Template Reference</a>	592
4.79.1	Detailed Description	593
4.79.2	Member Function Documentation	593
4.79.3	Member Data Documentation	593
4.80	<a href="#">__gnu_parallel::__fill_selector&lt; _It &gt; Struct Template Reference</a>	594
4.80.1	Detailed Description	594
4.80.2	Member Function Documentation	594
4.80.3	Member Data Documentation	594
4.81	<a href="#">__gnu_parallel::__find_first_of_selector&lt; _Filterator &gt; Struct Template Reference</a>	595
4.81.1	Detailed Description	595
4.81.2	Member Function Documentation	596
4.82	<a href="#">__gnu_parallel::__find_if_selector Struct Reference</a>	596
4.82.1	Detailed Description	597
4.82.2	Member Function Documentation	597
4.83	<a href="#">__gnu_parallel::__for_each_selector&lt; _It &gt; Struct Template Reference</a>	598
4.83.1	Detailed Description	598
4.83.2	Member Function Documentation	598
4.83.3	Member Data Documentation	598
4.84	<a href="#">__gnu_parallel::__generate_selector&lt; _It &gt; Struct Template Reference</a>	599
4.84.1	Detailed Description	599
4.84.2	Member Function Documentation	599
4.84.3	Member Data Documentation	601
4.85	<a href="#">__gnu_parallel::__generic_find_selector Struct Reference</a>	601
4.85.1	Detailed Description	601
4.86	<a href="#">__gnu_parallel::__generic_for_each_selector&lt; _It &gt; Struct Template Reference</a>	602

4.86.1 Detailed Description . . . . .	603
4.86.2 Member Data Documentation . . . . .	603
4.87 <code>__gnu_parallel::__identity_selector&lt;_It&gt;</code> Struct Template Reference . . . . .	603
4.87.1 Detailed Description . . . . .	604
4.87.2 Member Function Documentation . . . . .	604
4.87.3 Member Data Documentation . . . . .	604
4.88 <code>__gnu_parallel::__inner_product_selector&lt;_It, _It2, _Tp&gt;</code> Struct Template Reference . . . . .	605
4.88.1 Detailed Description . . . . .	605
4.88.2 Constructor & Destructor Documentation . . . . .	605
4.88.3 Member Function Documentation . . . . .	606
4.88.4 Member Data Documentation . . . . .	606
4.89 <code>__gnu_parallel::__max_element_reduct&lt;_Compare, _It&gt;</code> Struct Template Reference . . . . .	607
4.89.1 Detailed Description . . . . .	607
4.90 <code>__gnu_parallel::__min_element_reduct&lt;_Compare, _It&gt;</code> Struct Template Reference . . . . .	607
4.90.1 Detailed Description . . . . .	607
4.91 <code>__gnu_parallel::__mismatch_selector</code> Struct Reference . . . . .	608
4.91.1 Detailed Description . . . . .	608
4.91.2 Member Function Documentation . . . . .	608
4.92 <code>__gnu_parallel::__multiway_merge_3_variant_sentinel_switch&lt;__sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare&gt;</code> Struct Template Reference . . . . .	609
4.92.1 Detailed Description . . . . .	609
4.93 <code>__gnu_parallel::__multiway_merge_3_variant_sentinel_switch&lt;true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare&gt;</code> Struct Template Reference . . . . .	609
4.93.1 Detailed Description . . . . .	609
4.94 <code>__gnu_parallel::__multiway_merge_4_variant_sentinel_switch&lt;__sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare&gt;</code> Struct Template Reference . . . . .	610
4.94.1 Detailed Description . . . . .	610
4.95 <code>__gnu_parallel::__multiway_merge_4_variant_sentinel_switch&lt;true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare&gt;</code> Struct Template Reference . . . . .	610
4.95.1 Detailed Description . . . . .	610
4.96 <code>__gnu_parallel::__multiway_merge_k_variant_sentinel_switch&lt;__sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare&gt;</code> Struct Template Reference . . . . .	610
4.96.1 Detailed Description . . . . .	611
4.97 <code>__gnu_parallel::__multiway_merge_k_variant_sentinel_switch&lt;false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare&gt;</code> Struct Template Reference . . . . .	611
4.97.1 Detailed Description . . . . .	611
4.98 <code>__gnu_parallel::__replace_if_selector&lt;_It, _Op, _Tp&gt;</code> Struct Template Reference . . . . .	612
4.98.1 Detailed Description . . . . .	612
4.98.2 Constructor & Destructor Documentation . . . . .	612

4.98.3	Member Function Documentation	613
4.98.4	Member Data Documentation	614
4.99	<a href="#">__gnu_parallel::__replace_selector&lt;_It,_Tp&gt; Struct Template Reference</a>	614
4.99.1	Detailed Description	615
4.99.2	Constructor & Destructor Documentation	615
4.99.3	Member Function Documentation	615
4.99.4	Member Data Documentation	615
4.100	<a href="#">__gnu_parallel::__transform1_selector&lt;_It&gt; Struct Template Reference</a>	616
4.100.1	Detailed Description	616
4.100.2	Member Function Documentation	616
4.100.3	Member Data Documentation	617
4.101	<a href="#">__gnu_parallel::__transform2_selector&lt;_It&gt; Struct Template Reference</a>	617
4.101.1	Detailed Description	617
4.101.2	Member Function Documentation	618
4.101.3	Member Data Documentation	618
4.102	<a href="#">__gnu_parallel::__unary_negate&lt;_Predicate, argument_type&gt; Class Template Reference</a>	618
4.102.1	Detailed Description	619
4.102.2	Member Typedef Documentation	619
4.103	<a href="#">__gnu_parallel::_DRandomShufflingGlobalData&lt;_RAIter&gt; Struct Template Reference</a>	619
4.103.1	Detailed Description	620
4.103.2	Constructor & Destructor Documentation	620
4.103.3	Member Data Documentation	620
4.104	<a href="#">__gnu_parallel::_DRSSorterPU&lt;_RAIter,_RandomNumberGenerator&gt; Struct Template Reference</a>	621
4.104.1	Detailed Description	621
4.104.2	Member Data Documentation	622
4.105	<a href="#">__gnu_parallel::_DummyReduct Struct Reference</a>	622
4.105.1	Detailed Description	623
4.106	<a href="#">__gnu_parallel::_EqualFromLess&lt;_T1,_T2,_Compare&gt; Class Template Reference</a>	623
4.106.1	Detailed Description	623
4.106.2	Member Typedef Documentation	624
4.107	<a href="#">__gnu_parallel::_EqualTo&lt;_T1,_T2&gt; Struct Template Reference</a>	624
4.107.1	Detailed Description	625
4.107.2	Member Typedef Documentation	625
4.108	<a href="#">__gnu_parallel::_GuardedIterator&lt;_RAIter,_Compare&gt; Class Template Reference</a>	625
4.108.1	Detailed Description	625
4.108.2	Constructor & Destructor Documentation	626
4.108.3	Member Function Documentation	626

4.108.4 Friends And Related Function Documentation . . . . .	627
4.109 <code>__gnu_parallel::_IteratorPair&lt; _Iterator1, _Iterator2, _IteratorCategory &gt;</code> Class Template Reference . . . . .	628
4.109.1 Detailed Description . . . . .	629
4.109.2 Member Typedef Documentation . . . . .	629
4.109.3 Member Data Documentation . . . . .	629
4.110 <code>__gnu_parallel::_IteratorTriple&lt; _Iterator1, _Iterator2, _Iterator3, _IteratorCategory &gt;</code> Class Template Reference . . . . .	630
4.110.1 Detailed Description . . . . .	630
4.111 <code>__gnu_parallel::_Job&lt; _DifferenceTp &gt;</code> Struct Template Reference . . . . .	631
4.111.1 Detailed Description . . . . .	631
4.111.2 Member Data Documentation . . . . .	631
4.112 <code>__gnu_parallel::_Less&lt; _T1, _T2 &gt;</code> Struct Template Reference . . . . .	632
4.112.1 Detailed Description . . . . .	632
4.112.2 Member Typedef Documentation . . . . .	632
4.113 <code>__gnu_parallel::_Lexicographic&lt; _T1, _T2, _Compare &gt;</code> Class Template Reference . . . . .	633
4.113.1 Detailed Description . . . . .	633
4.113.2 Member Typedef Documentation . . . . .	634
4.114 <code>__gnu_parallel::_LexicographicReverse&lt; _T1, _T2, _Compare &gt;</code> Class Template Reference . . . . .	634
4.114.1 Detailed Description . . . . .	635
4.114.2 Member Typedef Documentation . . . . .	635
4.115 <code>__gnu_parallel::_LoserTree&lt; __stable, _Tp, _Compare &gt;</code> Class Template Reference . . . . .	636
4.115.1 Detailed Description . . . . .	636
4.115.2 Member Function Documentation . . . . .	637
4.115.3 Member Data Documentation . . . . .	637
4.116 <code>__gnu_parallel::_LoserTree&lt; false, _Tp, _Compare &gt;</code> Class Template Reference . . . . .	638
4.116.1 Detailed Description . . . . .	639
4.116.2 Member Function Documentation . . . . .	639
4.116.3 Member Data Documentation . . . . .	641
4.117 <code>__gnu_parallel::_LoserTreeBase&lt; _Tp, _Compare &gt;</code> Class Template Reference . . . . .	642
4.117.1 Detailed Description . . . . .	642
4.117.2 Constructor & Destructor Documentation . . . . .	643
4.117.3 Member Function Documentation . . . . .	643
4.117.4 Member Data Documentation . . . . .	644
4.118 <code>__gnu_parallel::_LoserTreeBase&lt; _Tp, _Compare &gt;::_Loser</code> Struct Reference . . . . .	644
4.118.1 Detailed Description . . . . .	644
4.118.2 Member Data Documentation . . . . .	645
4.119 <code>__gnu_parallel::_LoserTreePointer&lt; __stable, _Tp, _Compare &gt;</code> Class Template Reference . . . . .	646

4.119.1 Detailed Description . . . . .	646
4.120 <code>__gnu_parallel::_LoserTreePointer&lt; false, _Tp, _Compare &gt;</code> Class Template Reference . . . . .	647
4.120.1 Detailed Description . . . . .	647
4.121 <code>__gnu_parallel::_LoserTreePointerBase&lt; _Tp, _Compare &gt;</code> Class Template Reference . . . . .	648
4.121.1 Detailed Description . . . . .	648
4.122 <code>__gnu_parallel::_LoserTreePointerBase&lt; _Tp, _Compare &gt;::_Loser</code> Struct Reference . . . . .	648
4.122.1 Detailed Description . . . . .	649
4.123 <code>__gnu_parallel::_LoserTreePointerUnguarded&lt; __stable, _Tp, _Compare &gt;</code> Class Template Reference . . . . .	649
4.123.1 Detailed Description . . . . .	650
4.124 <code>__gnu_parallel::_LoserTreePointerUnguarded&lt; false, _Tp, _Compare &gt;</code> Class Template Reference . . . . .	650
4.124.1 Detailed Description . . . . .	651
4.125 <code>__gnu_parallel::_LoserTreePointerUnguardedBase&lt; _Tp, _Compare &gt;</code> Class Template Reference . . . . .	651
4.125.1 Detailed Description . . . . .	652
4.126 <code>__gnu_parallel::_LoserTreeTraits&lt; _Tp &gt;</code> Struct Template Reference . . . . .	652
4.126.1 Detailed Description . . . . .	652
4.126.2 Member Data Documentation . . . . .	652
4.127 <code>__gnu_parallel::_LoserTreeUnguarded&lt; __stable, _Tp, _Compare &gt;</code> Class Template Reference . . . . .	653
4.127.1 Detailed Description . . . . .	653
4.128 <code>__gnu_parallel::_LoserTreeUnguarded&lt; false, _Tp, _Compare &gt;</code> Class Template Reference . . . . .	654
4.128.1 Detailed Description . . . . .	654
4.129 <code>__gnu_parallel::_LoserTreeUnguardedBase&lt; _Tp, _Compare &gt;</code> Class Template Reference . . . . .	655
4.129.1 Detailed Description . . . . .	655
4.130 <code>__gnu_parallel::_Multiplies&lt; _Tp1, _Tp2, _Result &gt;</code> Struct Template Reference . . . . .	656
4.130.1 Detailed Description . . . . .	656
4.130.2 Member Typedef Documentation . . . . .	656
4.131 <code>__gnu_parallel::_Nothing</code> Struct Reference . . . . .	657
4.131.1 Detailed Description . . . . .	657
4.131.2 Member Function Documentation . . . . .	657
4.132 <code>__gnu_parallel::_Piece&lt; _DifferenceTp &gt;</code> Struct Template Reference . . . . .	657
4.132.1 Detailed Description . . . . .	658
4.132.2 Member Data Documentation . . . . .	658
4.133 <code>__gnu_parallel::_Plus&lt; _Tp1, _Tp2, _Result &gt;</code> Struct Template Reference . . . . .	658
4.133.1 Detailed Description . . . . .	659
4.133.2 Member Typedef Documentation . . . . .	659
4.134 <code>__gnu_parallel::_PMWMSortingData&lt; _RAIter &gt;</code> Struct Template Reference . . . . .	659
4.134.1 Detailed Description . . . . .	660
4.134.2 Member Data Documentation . . . . .	660

4.135 <a href="#">__gnu_parallel::_PseudoSequence&lt; _Tp, _DifferenceTp &gt; Class Template Reference</a> . . . . .	661
4.135.1 Detailed Description . . . . .	661
4.135.2 Constructor & Destructor Documentation . . . . .	661
4.135.3 Member Function Documentation . . . . .	662
4.136 <a href="#">__gnu_parallel::_PseudoSequenceIterator&lt; _Tp, _DifferenceTp &gt; Class Template Reference</a> . . . . .	662
4.136.1 Detailed Description . . . . .	662
4.137 <a href="#">__gnu_parallel::_QSBThreadLocal&lt; _RAIter &gt; Struct Template Reference</a> . . . . .	663
4.137.1 Detailed Description . . . . .	663
4.137.2 Member Typedef Documentation . . . . .	663
4.137.3 Constructor & Destructor Documentation . . . . .	663
4.137.4 Member Data Documentation . . . . .	664
4.138 <a href="#">__gnu_parallel::_RandomNumber Class Reference</a> . . . . .	664
4.138.1 Detailed Description . . . . .	665
4.138.2 Constructor & Destructor Documentation . . . . .	665
4.138.3 Member Function Documentation . . . . .	665
4.139 <a href="#">__gnu_parallel::_RestrictedBoundedConcurrentQueue&lt; _Tp &gt; Class Template Reference</a> . . . . .	666
4.139.1 Detailed Description . . . . .	666
4.139.2 Constructor & Destructor Documentation . . . . .	666
4.139.3 Member Function Documentation . . . . .	666
4.140 <a href="#">__gnu_parallel::_SamplingSorter&lt; __stable, _RAIter, _StrictWeakOrdering &gt; Struct Template Reference</a> . . . . .	667
4.140.1 Detailed Description . . . . .	667
4.141 <a href="#">__gnu_parallel::_SamplingSorter&lt; false, _RAIter, _StrictWeakOrdering &gt; Struct Template Reference</a> . . . . .	667
4.141.1 Detailed Description . . . . .	667
4.142 <a href="#">__gnu_parallel::_Settings Struct Reference</a> . . . . .	668
4.142.1 Detailed Description . . . . .	669
4.142.2 Member Function Documentation . . . . .	669
4.142.3 Member Data Documentation . . . . .	669
4.143 <a href="#">__gnu_parallel::_SplitConsistently&lt; __exact, _RAIter, _Compare, _SortingPlacesIterator &gt; Struct Template Reference</a> . . . . .	674
4.143.1 Detailed Description . . . . .	674
4.144 <a href="#">__gnu_parallel::_SplitConsistently&lt; false, _RAIter, _Compare, _SortingPlacesIterator &gt; Struct Template Reference</a> . . . . .	674
4.144.1 Detailed Description . . . . .	674
4.145 <a href="#">__gnu_parallel::_SplitConsistently&lt; true, _RAIter, _Compare, _SortingPlacesIterator &gt; Struct Template Reference</a> . . . . .	675
4.145.1 Detailed Description . . . . .	675
4.146 <a href="#">__gnu_parallel::balanced_quicksort_tag Struct Reference</a> . . . . .	675
4.146.1 Detailed Description . . . . .	676

4.146.2 Member Function Documentation	676
4.147 <a href="#">__gnu_parallel::balanced_tag</a> Struct Reference	676
4.147.1 Detailed Description	677
4.147.2 Member Function Documentation	677
4.148 <a href="#">__gnu_parallel::constant_size_blocks_tag</a> Struct Reference	677
4.148.1 Detailed Description	678
4.149 <a href="#">__gnu_parallel::default_parallel_tag</a> Struct Reference	678
4.149.1 Detailed Description	678
4.149.2 Member Function Documentation	678
4.150 <a href="#">__gnu_parallel::equal_split_tag</a> Struct Reference	679
4.150.1 Detailed Description	679
4.151 <a href="#">__gnu_parallel::exact_tag</a> Struct Reference	680
4.151.1 Detailed Description	680
4.151.2 Member Function Documentation	680
4.152 <a href="#">__gnu_parallel::find_tag</a> Struct Reference	681
4.152.1 Detailed Description	681
4.153 <a href="#">__gnu_parallel::growing_blocks_tag</a> Struct Reference	682
4.153.1 Detailed Description	682
4.154 <a href="#">__gnu_parallel::multiway_mergesort_exact_tag</a> Struct Reference	682
4.154.1 Detailed Description	683
4.154.2 Member Function Documentation	683
4.155 <a href="#">__gnu_parallel::multiway_mergesort_sampling_tag</a> Struct Reference	684
4.155.1 Detailed Description	684
4.155.2 Member Function Documentation	684
4.156 <a href="#">__gnu_parallel::multiway_mergesort_tag</a> Struct Reference	685
4.156.1 Detailed Description	685
4.156.2 Member Function Documentation	685
4.157 <a href="#">__gnu_parallel::omp_loop_static_tag</a> Struct Reference	687
4.157.1 Detailed Description	687
4.157.2 Member Function Documentation	687
4.158 <a href="#">__gnu_parallel::omp_loop_tag</a> Struct Reference	689
4.158.1 Detailed Description	689
4.158.2 Member Function Documentation	689
4.159 <a href="#">__gnu_parallel::parallel_tag</a> Struct Reference	692
4.159.1 Detailed Description	693
4.159.2 Constructor & Destructor Documentation	693
4.159.3 Member Function Documentation	693

4.160 <a href="#">__gnu_parallel::quicksort_tag</a> Struct Reference	694
4.160.1 Detailed Description	694
4.160.2 Member Function Documentation	694
4.161 <a href="#">__gnu_parallel::sampling_tag</a> Struct Reference	695
4.161.1 Detailed Description	695
4.161.2 Member Function Documentation	695
4.162 <a href="#">__gnu_parallel::sequential_tag</a> Struct Reference	697
4.162.1 Detailed Description	697
4.163 <a href="#">__gnu_parallel::unbalanced_tag</a> Struct Reference	697
4.163.1 Detailed Description	697
4.163.2 Member Function Documentation	698
4.164 <a href="#">__gnu_pbds::associative_tag</a> Struct Reference	698
4.164.1 Detailed Description	698
4.165 <a href="#">__gnu_pbds::basic_branch&lt; Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc &gt;</a> Class Template Reference	699
4.165.1 Detailed Description	699
4.166 <a href="#">__gnu_pbds::basic_branch_tag</a> Struct Reference	700
4.166.1 Detailed Description	700
4.167 <a href="#">__gnu_pbds::basic_hash_table&lt; Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc &gt;</a> Class Template Reference	700
4.167.1 Detailed Description	700
4.168 <a href="#">__gnu_pbds::basic_hash_tag</a> Struct Reference	702
4.168.1 Detailed Description	702
4.169 <a href="#">__gnu_pbds::basic_invalidation_guarantee</a> Struct Reference	703
4.169.1 Detailed Description	703
4.170 <a href="#">__gnu_pbds::binary_heap_tag</a> Struct Reference	704
4.170.1 Detailed Description	704
4.171 <a href="#">__gnu_pbds::binomial_heap_tag</a> Struct Reference	705
4.171.1 Detailed Description	705
4.172 <a href="#">__gnu_pbds::cc_hash_max_collision_check_resize_trigger&lt; External_Load_Access, Size_Type &gt;</a> Class Template Reference	705
4.172.1 Detailed Description	706
4.172.2 Member Enumeration Documentation	706
4.172.3 Constructor & Destructor Documentation	706
4.172.4 Member Function Documentation	707
4.173 <a href="#">__gnu_pbds::cc_hash_table&lt; Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc &gt;</a> Class Template Reference	709
4.173.1 Detailed Description	710



4.173.2 Constructor & Destructor Documentation . . . . .	711
4.174 __gnu_pbds::cc_hash_tag Struct Reference . . . . .	714
4.174.1 Detailed Description . . . . .	714
4.175 __gnu_pbds::container_error Struct Reference . . . . .	715
4.175.1 Detailed Description . . . . .	715
4.176 __gnu_pbds::container_tag Struct Reference . . . . .	715
4.176.1 Detailed Description . . . . .	716
4.177 __gnu_pbds::container_traits< Cntnr > Struct Template Reference . . . . .	716
4.177.1 Detailed Description . . . . .	716
4.178 __gnu_pbds::container_traits_base< _Tag > Struct Template Reference . . . . .	717
4.178.1 Detailed Description . . . . .	717
4.179 __gnu_pbds::container_traits_base< binary_heap_tag > Struct Template Reference . . . . .	717
4.179.1 Detailed Description . . . . .	717
4.180 __gnu_pbds::container_traits_base< binomial_heap_tag > Struct Template Reference . . . . .	717
4.180.1 Detailed Description . . . . .	717
4.181 __gnu_pbds::container_traits_base< cc_hash_tag > Struct Template Reference . . . . .	718
4.181.1 Detailed Description . . . . .	718
4.182 __gnu_pbds::container_traits_base< gp_hash_tag > Struct Template Reference . . . . .	718
4.182.1 Detailed Description . . . . .	718
4.183 __gnu_pbds::container_traits_base< list_update_tag > Struct Template Reference . . . . .	718
4.183.1 Detailed Description . . . . .	719
4.184 __gnu_pbds::container_traits_base< ov_tree_tag > Struct Template Reference . . . . .	719
4.184.1 Detailed Description . . . . .	719
4.185 __gnu_pbds::container_traits_base< pairing_heap_tag > Struct Template Reference . . . . .	719
4.185.1 Detailed Description . . . . .	719
4.186 __gnu_pbds::container_traits_base< pat_trie_tag > Struct Template Reference . . . . .	720
4.186.1 Detailed Description . . . . .	720
4.187 __gnu_pbds::container_traits_base< rb_tree_tag > Struct Template Reference . . . . .	720
4.187.1 Detailed Description . . . . .	720
4.188 __gnu_pbds::container_traits_base< rc_binomial_heap_tag > Struct Template Reference . . . . .	720
4.188.1 Detailed Description . . . . .	721
4.189 __gnu_pbds::container_traits_base< splay_tree_tag > Struct Template Reference . . . . .	721
4.189.1 Detailed Description . . . . .	721
4.190 __gnu_pbds::container_traits_base< thin_heap_tag > Struct Template Reference . . . . .	721
4.190.1 Detailed Description . . . . .	721
4.191 __gnu_pbds::detail::bin_search_tree_const_it< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > Class Template Reference . . . . .	722

4.191.1 Detailed Description	723
4.192 <code>__gnu_pbds::detail::bin_search_tree_const_node_it_&lt; Node, Const_Iterator, Iterator, _Alloc &gt; Class Template Reference</code>	723
4.192.1 Detailed Description	724
4.192.2 Member Typedef Documentation	724
4.192.3 Member Function Documentation	725
4.193 <code>__gnu_pbds::detail::bin_search_tree_it_&lt; Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc &gt; Class Template Reference</code>	726
4.193.1 Detailed Description	728
4.194 <code>__gnu_pbds::detail::bin_search_tree_node_it_&lt; Node, Const_Iterator, Iterator, _Alloc &gt; Class Template Reference</code>	728
4.194.1 Detailed Description	729
4.194.2 Member Typedef Documentation	729
4.194.3 Member Function Documentation	730
4.195 <code>__gnu_pbds::detail::bin_search_tree_traits&lt; Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc &gt; Struct Template Reference</code>	731
4.195.1 Detailed Description	731
4.195.2 Member Typedef Documentation	732
4.196 <code>__gnu_pbds::detail::bin_search_tree_traits&lt; Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc &gt; Struct Template Reference</code>	732
4.196.1 Detailed Description	732
4.196.2 Member Typedef Documentation	732
4.197 <code>__gnu_pbds::detail::binary_heap&lt; Value_Type, Cmp_Fn, _Alloc &gt; Class Template Reference</code>	733
4.197.1 Detailed Description	735
4.198 <code>__gnu_pbds::detail::binary_heap_const_iterator_&lt; Value_Type, Entry, Simple, _Alloc &gt; Class Template Reference</code>	735
4.198.1 Detailed Description	736
4.198.2 Member Typedef Documentation	736
4.198.3 Constructor & Destructor Documentation	737
4.198.4 Member Function Documentation	737
4.199 <code>__gnu_pbds::detail::binary_heap_point_const_iterator_&lt; Value_Type, Entry, Simple, _Alloc &gt; Class Template Reference</code>	738
4.199.1 Detailed Description	739
4.199.2 Member Typedef Documentation	739
4.199.3 Constructor & Destructor Documentation	740
4.199.4 Member Function Documentation	741
4.200 <code>__gnu_pbds::detail::binomial_heap&lt; Value_Type, Cmp_Fn, _Alloc &gt; Class Template Reference</code>	741
4.200.1 Detailed Description	743
4.201 <code>__gnu_pbds::detail::binomial_heap_base&lt; Value_Type, Cmp_Fn, _Alloc &gt; Class Template Reference</code>	744

4.201.1 Detailed Description	. 745
4.202 <code>__gnu_pbds::detail::branch_policy&lt; Node_Cltr, Node_Itr, _Alloc &gt;</code> Struct Template Reference	. 746
4.202.1 Detailed Description	. 746
4.203 <code>__gnu_pbds::detail::branch_policy&lt; Node_Cltr, Node_Cltr, _Alloc &gt;</code> Struct Template Reference	. 747
4.203.1 Detailed Description	. 747
4.204 <code>__gnu_pbds::detail::cc_ht_map&lt; Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy &gt;</code> Class Template Reference	. 748
4.204.1 Detailed Description	. 750
4.204.2 Member Function Documentation	. 751
4.205 <code>__gnu_pbds::detail::cond_dealtor&lt; Entry, _Alloc &gt;</code> Class Template Reference	. 752
4.205.1 Detailed Description	. 753
4.206 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, Mapped, _Alloc, Tag, Policy_TI &gt;</code> Struct Template Reference	. 753
4.206.1 Detailed Description	. 753
4.207 <code>__gnu_pbds::detail::container_base_dispatch&lt; _VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type &gt;</code> Struct Template Reference	. 753
4.207.1 Detailed Description	. 753
4.207.2 Member Typedef Documentation	. 754
4.208 <code>__gnu_pbds::detail::container_base_dispatch&lt; _VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type &gt;</code> Struct Template Reference	. 754
4.208.1 Detailed Description	. 754
4.208.2 Member Typedef Documentation	. 754
4.209 <code>__gnu_pbds::detail::container_base_dispatch&lt; _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type &gt;</code> Struct Template Reference	. 754
4.209.1 Detailed Description	. 755
4.209.2 Member Typedef Documentation	. 755
4.210 <code>__gnu_pbds::detail::container_base_dispatch&lt; _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type &gt;</code> Struct Template Reference	. 755
4.210.1 Detailed Description	. 755
4.210.2 Member Typedef Documentation	. 755
4.211 <code>__gnu_pbds::detail::container_base_dispatch&lt; _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type &gt;</code> Struct Template Reference	. 756
4.211.1 Detailed Description	. 756
4.211.2 Member Typedef Documentation	. 756
4.212 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, Mapped, _Alloc, cc_hash_tag, Policy_TI &gt;</code> Struct Template Reference	. 756
4.212.1 Detailed Description	. 756
4.212.2 Member Typedef Documentation	. 756
4.213 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, Mapped, _Alloc, gp_hash_tag, Policy_TI &gt;</code> Struct Template Reference	. 757

4.213.1 Detailed Description	. 757
4.213.2 Member Typedef Documentation	. 757
4.214 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, Mapped, _Alloc, list_update_tag, Policy_TI &gt;</code> Struct Template Reference	. 757
4.214.1 Detailed Description	. 757
4.214.2 Member Typedef Documentation	. 758
4.215 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, Mapped, _Alloc, ov_tree_tag, Policy_TI &gt;</code> Struct Template Reference	. 758
4.215.1 Detailed Description	. 758
4.215.2 Member Typedef Documentation	. 758
4.216 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, Mapped, _Alloc, pat_trie_tag, Policy_TI &gt;</code> Struct Template Reference	. 758
4.216.1 Detailed Description	. 759
4.217 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, Mapped, _Alloc, rb_tree_tag, Policy_TI &gt;</code> Struct Template Reference	. 759
4.217.1 Detailed Description	. 759
4.217.2 Member Typedef Documentation	. 759
4.218 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, Mapped, _Alloc, splay_tree_tag, Policy_TI &gt;</code> Struct Template Reference	. 759
4.218.1 Detailed Description	. 760
4.218.2 Member Typedef Documentation	. 760
4.219 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, null_type, _Alloc, cc_hash_tag, Policy_TI &gt;</code> Struct Template Reference	. 760
4.219.1 Detailed Description	. 760
4.219.2 Member Typedef Documentation	. 760
4.220 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, null_type, _Alloc, gp_hash_tag, Policy_TI &gt;</code> Struct Template Reference	. 761
4.220.1 Detailed Description	. 761
4.220.2 Member Typedef Documentation	. 761
4.221 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, null_type, _Alloc, list_update_tag, Policy_TI &gt;</code> Struct Template Reference	. 761
4.221.1 Detailed Description	. 761
4.221.2 Member Typedef Documentation	. 761
4.222 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, null_type, _Alloc, ov_tree_tag, Policy_TI &gt;</code> Struct Template Reference	. 762
4.222.1 Detailed Description	. 762
4.222.2 Member Typedef Documentation	. 762
4.223 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, null_type, _Alloc, pat_trie_tag, Policy_TI &gt;</code> Struct Template Reference	. 762
4.223.1 Detailed Description	. 762

4.223.2 Member Typedef Documentation . . . . .	763
4.224 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, null_type, _Alloc, rb_tree_tag, Policy_Tl &gt; Struct Template Reference</code> . . . . .	763
4.224.1 Detailed Description . . . . .	763
4.225 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, null_type, _Alloc, splay_tree_tag, Policy_Tl &gt; Struct Template Reference</code> . . . . .	763
4.225.1 Detailed Description . . . . .	763
4.225.2 Member Typedef Documentation . . . . .	763
4.226 <code>__gnu_pbds::detail::default_comb_hash_fn Struct Reference</code> . . . . .	764
4.226.1 Detailed Description . . . . .	764
4.226.2 Member Typedef Documentation . . . . .	764
4.227 <code>__gnu_pbds::detail::default_eq_fn&lt; Key &gt; Struct Template Reference</code> . . . . .	764
4.227.1 Detailed Description . . . . .	764
4.227.2 Member Typedef Documentation . . . . .	765
4.228 <code>__gnu_pbds::detail::default_hash_fn&lt; Key &gt; Struct Template Reference</code> . . . . .	765
4.228.1 Detailed Description . . . . .	765
4.228.2 Member Typedef Documentation . . . . .	765
4.229 <code>__gnu_pbds::detail::default_probe_fn&lt; Comb_Probe_Fn &gt; Struct Template Reference</code> . . . . .	765
4.229.1 Detailed Description . . . . .	765
4.229.2 Member Typedef Documentation . . . . .	766
4.230 <code>__gnu_pbds::detail::default_resize_policy&lt; Comb_Hash_Fn &gt; Struct Template Reference</code> . . . . .	766
4.230.1 Detailed Description . . . . .	766
4.230.2 Member Typedef Documentation . . . . .	766
4.231 <code>__gnu_pbds::detail::default_trie_access_traits&lt; Key &gt; Struct Template Reference</code> . . . . .	766
4.231.1 Detailed Description . . . . .	766
4.232 <code>__gnu_pbds::detail::default_trie_access_traits&lt; std::basic_string&lt; Char, Char_Traits, std::allocator&lt; char &gt; &gt; &gt; Struct Template Reference</code> . . . . .	767
4.232.1 Detailed Description . . . . .	767
4.232.2 Member Typedef Documentation . . . . .	767
4.233 <code>__gnu_pbds::detail::default_update_policy Struct Reference</code> . . . . .	767
4.233.1 Detailed Description . . . . .	767
4.233.2 Member Typedef Documentation . . . . .	767
4.234 <code>__gnu_pbds::detail::dumnode_const_iterator&lt; Key, Data, _Alloc &gt; Struct Template Reference</code> . . . . .	768
4.234.1 Detailed Description . . . . .	768
4.235 <code>__gnu_pbds::detail::entry_cmp&lt; _VTp, Cmp_Fn, _Alloc, No_Throw &gt; Struct Template Reference</code> . . . . .	768
4.235.1 Detailed Description . . . . .	768
4.236 <code>__gnu_pbds::detail::entry_cmp&lt; _VTp, Cmp_Fn, _Alloc, false &gt; Struct Template Reference</code> . . . . .	768
4.236.1 Detailed Description . . . . .	768

4.237 <code>__gnu_pbds::detail::entry_cmp&lt;_VTp, Cmp_Fn, _Alloc, false &gt;::type</code> Struct Reference . . . . .	769
4.237.1 Detailed Description . . . . .	769
4.238 <code>__gnu_pbds::detail::entry_cmp&lt;_VTp, Cmp_Fn, _Alloc, true &gt;</code> Struct Template Reference . . . . .	769
4.238.1 Detailed Description . . . . .	769
4.238.2 Member Typedef Documentation . . . . .	769
4.239 <code>__gnu_pbds::detail::entry_pred&lt;_VTp, Pred, _Alloc, No_Throw &gt;</code> Struct Template Reference . . . . .	770
4.239.1 Detailed Description . . . . .	770
4.240 <code>__gnu_pbds::detail::entry_pred&lt;_VTp, Pred, _Alloc, false &gt;</code> Struct Template Reference . . . . .	770
4.240.1 Detailed Description . . . . .	770
4.241 <code>__gnu_pbds::detail::entry_pred&lt;_VTp, Pred, _Alloc, true &gt;</code> Struct Template Reference . . . . .	770
4.241.1 Detailed Description . . . . .	770
4.242 <code>__gnu_pbds::detail::eq_by_less&lt;Key, Cmp_Fn &gt;</code> Struct Template Reference . . . . .	771
4.242.1 Detailed Description . . . . .	771
4.243 <code>__gnu_pbds::detail::gp_ht_map&lt;Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy &gt;</code> Class Template Reference . . . . .	771
4.243.1 Detailed Description . . . . .	773
4.243.2 Member Function Documentation . . . . .	774
4.244 <code>__gnu_pbds::detail::hash_eq_fn&lt;Key, Eq_Fn, _Alloc, Store_Hash &gt;</code> Struct Template Reference . . . . .	776
4.244.1 Detailed Description . . . . .	776
4.245 <code>__gnu_pbds::detail::hash_eq_fn&lt;Key, Eq_Fn, _Alloc, false &gt;</code> Struct Template Reference . . . . .	776
4.245.1 Detailed Description . . . . .	777
4.246 <code>__gnu_pbds::detail::hash_eq_fn&lt;Key, Eq_Fn, _Alloc, true &gt;</code> Struct Template Reference . . . . .	777
4.246.1 Detailed Description . . . . .	777
4.247 <code>__gnu_pbds::detail::hash_load_check_resize_trigger_size_base&lt;Size_Type, Hold_Size &gt;</code> Class Template Reference . . . . .	777
4.247.1 Detailed Description . . . . .	777
4.248 <code>__gnu_pbds::detail::hash_load_check_resize_trigger_size_base&lt;Size_Type, true &gt;</code> Class Template Reference . . . . .	778
4.248.1 Detailed Description . . . . .	778
4.249 <code>__gnu_pbds::detail::left_child_next_sibling_heap&lt;Value_Type, Cmp_Fn, Node_Metadata, _Alloc &gt;</code> Class Template Reference . . . . .	778
4.249.1 Detailed Description . . . . .	780
4.250 <code>__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator&lt;Node, _Alloc &gt;</code> Class Template Reference . . . . .	780
4.250.1 Detailed Description . . . . .	781
4.250.2 Member Typedef Documentation . . . . .	781
4.250.3 Constructor & Destructor Documentation . . . . .	782
4.250.4 Member Function Documentation . . . . .	782

4.251	<a href="#">__gnu_pbds::detail::left_child_next_sibling_heap_node_&lt; _Value, _Metadata, _Alloc &gt; Struct Template Reference</a>	783
4.251.1	Detailed Description	783
4.252	<a href="#">__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_&lt; Node, _Alloc &gt; Class Template Reference</a>	784
4.252.1	Detailed Description	785
4.252.2	Member Typedef Documentation	785
4.252.3	Constructor & Destructor Documentation	786
4.252.4	Member Function Documentation	786
4.253	<a href="#">__gnu_pbds::detail::lu_counter_metadata&lt; Size_Type &gt; Class Template Reference</a>	787
4.253.1	Detailed Description	787
4.254	<a href="#">__gnu_pbds::detail::lu_counter_policy_base&lt; Size_Type &gt; Class Template Reference</a>	787
4.254.1	Detailed Description	787
4.255	<a href="#">__gnu_pbds::detail::lu_map&lt; Key, Mapped, Eq_Fn, _Alloc, Update_Policy &gt; Class Template Reference</a>	788
4.255.1	Detailed Description	790
4.256	<a href="#">__gnu_pbds::detail::mask_based_range_hashing&lt; Size_Type &gt; Class Template Reference</a>	790
4.256.1	Detailed Description	791
4.257	<a href="#">__gnu_pbds::detail::mod_based_range_hashing&lt; Size_Type &gt; Class Template Reference</a>	791
4.257.1	Detailed Description	791
4.258	<a href="#">__gnu_pbds::detail::no_throw_copies&lt; Key, Mapped &gt; Struct Template Reference</a>	792
4.258.1	Detailed Description	792
4.259	<a href="#">__gnu_pbds::detail::no_throw_copies&lt; Key, null_type &gt; Struct Template Reference</a>	792
4.259.1	Detailed Description	792
4.260	<a href="#">__gnu_pbds::detail::ov_tree_map&lt; Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc &gt; Class Template Reference</a>	793
4.260.1	Detailed Description	795
4.260.2	Member Function Documentation	795
4.261	<a href="#">__gnu_pbds::detail::ov_tree_map&lt; Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc &gt;::cond_dtor&lt; Size_Type &gt; Class Template Reference</a>	796
4.261.1	Detailed Description	796
4.262	<a href="#">__gnu_pbds::detail::ov_tree_node_const_it_&lt; Value_Type, Metadata_Type, _Alloc &gt; Class Template Reference</a>	796
4.262.1	Detailed Description	797
4.262.2	Member Function Documentation	798
4.263	<a href="#">__gnu_pbds::detail::ov_tree_node_it_&lt; Value_Type, Metadata_Type, _Alloc &gt; Class Template Reference</a>	798
4.263.1	Detailed Description	799
4.263.2	Member Function Documentation	799
4.264	<a href="#">__gnu_pbds::detail::pairing_heap&lt; Value_Type, Cmp_Fn, _Alloc &gt; Class Template Reference</a>	800

4.264.1 Detailed Description	802
4.265 <code>__gnu_pbds::detail::pat_trie_base</code> Struct Reference	802
4.265.1 Detailed Description	803
4.265.2 Member Enumeration Documentation	803
4.266 <code>__gnu_pbds::detail::pat_trie_base::_Clter&lt; Node, Leaf, Head, Inode, Is_Forward_Iterator &gt;</code> Class Template Reference	803
4.266.1 Detailed Description	805
4.267 <code>__gnu_pbds::detail::pat_trie_base::_Head&lt; _ATraits, Metadata &gt;</code> Struct Template Reference	805
4.267.1 Detailed Description	806
4.268 <code>__gnu_pbds::detail::pat_trie_base::_Inode&lt; _ATraits, Metadata &gt;</code> Struct Template Reference	806
4.268.1 Detailed Description	807
4.269 <code>__gnu_pbds::detail::pat_trie_base::_Inode&lt; _ATraits, Metadata &gt;::const_iterator</code> Struct Reference	808
4.269.1 Detailed Description	809
4.270 <code>__gnu_pbds::detail::pat_trie_base::_Inode&lt; _ATraits, Metadata &gt;::iterator</code> Struct Reference	809
4.270.1 Detailed Description	810
4.271 <code>__gnu_pbds::detail::pat_trie_base::_Iter&lt; Node, Leaf, Head, Inode, Is_Forward_Iterator &gt;</code> Class Template Reference	810
4.271.1 Detailed Description	812
4.272 <code>__gnu_pbds::detail::pat_trie_base::_Leaf&lt; _ATraits, Metadata &gt;</code> Struct Template Reference	812
4.272.1 Detailed Description	813
4.273 <code>__gnu_pbds::detail::pat_trie_base::_Metadata&lt; Metadata, _Alloc &gt;</code> Struct Template Reference	813
4.273.1 Detailed Description	813
4.274 <code>__gnu_pbds::detail::pat_trie_base::_Metadata&lt; null_type, _Alloc &gt;</code> Struct Template Reference	814
4.274.1 Detailed Description	814
4.275 <code>__gnu_pbds::detail::pat_trie_base::_Node_base&lt; _ATraits, Metadata &gt;</code> Struct Template Reference	814
4.275.1 Detailed Description	815
4.276 <code>__gnu_pbds::detail::pat_trie_base::_Node_citer&lt; Node, Leaf, Head, Inode, _Clterator, Iterator, _Alloc &gt;</code> Class Template Reference	815
4.276.1 Detailed Description	816
4.276.2 Member Typedef Documentation	817
4.276.3 Member Function Documentation	817
4.277 <code>__gnu_pbds::detail::pat_trie_base::_Node_iter&lt; Node, Leaf, Head, Inode, _Clterator, Iterator, _Alloc &gt;</code> Class Template Reference	818
4.277.1 Detailed Description	819
4.277.2 Member Typedef Documentation	820
4.277.3 Member Function Documentation	820
4.278 <code>__gnu_pbds::detail::pat_trie_map&lt; Key, Mapped, Node_And_It_Traits, _Alloc &gt;</code> Class Template Reference	821



4.278.1 Detailed Description	823
4.278.2 Member Enumeration Documentation	824
4.278.3 Member Function Documentation	824
4.279 <code>__gnu_pbds::detail::probe_fn_base&lt; _Alloc &gt;</code> Class Template Reference	824
4.279.1 Detailed Description	824
4.280 <code>__gnu_pbds::detail::ranged_hash_fn&lt; Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash &gt;</code> Class Template Reference	825
4.280.1 Detailed Description	825
4.281 <code>__gnu_pbds::detail::ranged_hash_fn&lt; Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false &gt;</code> Class Template Reference	825
4.281.1 Detailed Description	826
4.282 <code>__gnu_pbds::detail::ranged_hash_fn&lt; Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true &gt;</code> Class Template Reference	826
4.282.1 Detailed Description	827
4.283 <code>__gnu_pbds::detail::ranged_hash_fn&lt; Key, null_type, _Alloc, Comb_Hash_Fn, false &gt;</code> Class Template Reference	827
4.283.1 Detailed Description	827
4.284 <code>__gnu_pbds::detail::ranged_hash_fn&lt; Key, null_type, _Alloc, Comb_Hash_Fn, true &gt;</code> Class Template Reference	827
4.284.1 Detailed Description	828
4.285 <code>__gnu_pbds::detail::ranged_probe_fn&lt; Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash &gt;</code> Class Template Reference	828
4.285.1 Detailed Description	828
4.286 <code>__gnu_pbds::detail::ranged_probe_fn&lt; Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false &gt;</code> Class Template Reference	829
4.286.1 Detailed Description	829
4.287 <code>__gnu_pbds::detail::ranged_probe_fn&lt; Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true &gt;</code> Class Template Reference	829
4.287.1 Detailed Description	830
4.288 <code>__gnu_pbds::detail::ranged_probe_fn&lt; Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false &gt;</code> Class Template Reference	830
4.288.1 Detailed Description	831
4.289 <code>__gnu_pbds::detail::rb_tree_map&lt; Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc &gt;</code> Class Template Reference	831
4.289.1 Detailed Description	834
4.289.2 Member Function Documentation	834
4.290 <code>__gnu_pbds::detail::rb_tree_node_&lt; Value_Type, Metadata, _Alloc &gt;</code> Struct Template Reference	834
4.290.1 Detailed Description	835
4.291 <code>__gnu_pbds::detail::rc&lt; _Node, _Alloc &gt;</code> Class Template Reference	835
4.291.1 Detailed Description	836

4.292__gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference . . .	836
4.292.1 Detailed Description . . . . .	838
4.293__gnu_pbds::detail::resize_policy< _Tp > Class Template Reference . . . . .	838
4.293.1 Detailed Description . . . . .	839
4.294__gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > Class Template Reference . . . . .	839
4.294.1 Detailed Description . . . . .	842
4.294.2 Member Function Documentation . . . . .	842
4.295__gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc > Struct Template Reference . . .	842
4.295.1 Detailed Description . . . . .	843
4.296__gnu_pbds::detail::stored_data< _Tv, _Th > Struct Template Reference . . . . .	844
4.296.1 Detailed Description . . . . .	844
4.297__gnu_pbds::detail::stored_data< _Tv, null_type > Struct Template Reference . . . . .	845
4.297.1 Detailed Description . . . . .	845
4.298__gnu_pbds::detail::stored_hash< _Th > Struct Template Reference . . . . .	846
4.298.1 Detailed Description . . . . .	846
4.299__gnu_pbds::detail::stored_value< _Tv > Struct Template Reference . . . . .	847
4.299.1 Detailed Description . . . . .	847
4.300__gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits > Struct Template Reference . . .	847
4.300.1 Detailed Description . . . . .	848
4.301__gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference . . . . .	848
4.301.1 Detailed Description . . . . .	850
4.302__gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp > Struct Template Reference . . . . .	850
4.302.1 Detailed Description . . . . .	850
4.303__gnu_pbds::detail::tree_metadata_helper< Node_Update, false > Struct Template Reference . . . . .	850
4.303.1 Detailed Description . . . . .	851
4.304__gnu_pbds::detail::tree_metadata_helper< Node_Update, true > Struct Template Reference . . . . .	851
4.304.1 Detailed Description . . . . .	851
4.305__gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc > Struct Template Reference . . . . .	851
4.305.1 Detailed Description . . . . .	851
4.306__gnu_pbds::detail::tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc > Struct Template Reference . . . . .	852
4.306.1 Detailed Description . . . . .	852
4.307__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc > Struct Template Reference . . . . .	852
4.307.1 Detailed Description . . . . .	852
4.307.2 Member Typedef Documentation . . . . .	852

4.308__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc > Struct Template Reference	853
4.308.1 Detailed Description	853
4.308.2 Member Typedef Documentation	854
4.309__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc > Struct Template Reference	854
4.309.1 Detailed Description	855
4.309.2 Member Typedef Documentation	855
4.310__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc > Struct Template Reference	855
4.310.1 Detailed Description	856
4.310.2 Member Typedef Documentation	856
4.311__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc > Struct Template Reference	856
4.311.1 Detailed Description	857
4.311.2 Member Typedef Documentation	857
4.312__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc > Struct Template Reference	858
4.312.1 Detailed Description	858
4.312.2 Member Typedef Documentation	859
4.313__gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp > Struct Template Reference	859
4.313.1 Detailed Description	859
4.314__gnu_pbds::detail::trie_metadata_helper< Node_Update, false > Struct Template Reference	859
4.314.1 Detailed Description	859
4.315__gnu_pbds::detail::trie_metadata_helper< Node_Update, true > Struct Template Reference	860
4.315.1 Detailed Description	860
4.316__gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc > Struct Template Reference	860
4.316.1 Detailed Description	860
4.317__gnu_pbds::detail::trie_policy_base< Node_Cltr, Node_Itr, _ATraits, _Alloc > Class Template Reference	861
4.317.1 Detailed Description	862
4.318__gnu_pbds::detail::trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc > Struct Template Reference	862
4.318.1 Detailed Description	862
4.319__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc > Struct Template Reference	862
4.319.1 Detailed Description	863
4.319.2 Member Typedef Documentation	863
4.320__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc > Struct Template Reference	864

4.320.1 Detailed Description	864
4.320.2 Member Typedef Documentation	864
4.321 <code>__gnu_pbds::detail::type_base&lt; Key, Mapped, _Alloc, Store_Hash &gt;</code> Struct Template Reference	865
4.321.1 Detailed Description	865
4.322 <code>__gnu_pbds::detail::type_base&lt; Key, Mapped, _Alloc, false &gt;</code> Struct Template Reference	866
4.322.1 Detailed Description	866
4.323 <code>__gnu_pbds::detail::type_base&lt; Key, Mapped, _Alloc, true &gt;</code> Struct Template Reference	866
4.323.1 Detailed Description	867
4.324 <code>__gnu_pbds::detail::type_base&lt; Key, null_type, _Alloc, false &gt;</code> Struct Template Reference	867
4.324.1 Detailed Description	867
4.325 <code>__gnu_pbds::detail::type_base&lt; Key, null_type, _Alloc, true &gt;</code> Struct Template Reference	867
4.325.1 Detailed Description	868
4.326 <code>__gnu_pbds::detail::type_dispatch&lt; Key, Mapped, _Alloc, Store_Hash &gt;</code> Struct Template Reference	868
4.326.1 Detailed Description	868
4.327 <code>__gnu_pbds::detail::types_traits&lt; Key, Mapped, _Alloc, Store_Hash &gt;</code> Struct Template Reference	869
4.327.1 Detailed Description	869
4.328 <code>__gnu_pbds::direct_mask_range_hashing&lt; Size_Type &gt;</code> Class Template Reference	870
4.328.1 Detailed Description	870
4.328.2 Member Function Documentation	870
4.329 <code>__gnu_pbds::direct_mod_range_hashing&lt; Size_Type &gt;</code> Class Template Reference	871
4.329.1 Detailed Description	871
4.329.2 Member Function Documentation	872
4.330 <code>__gnu_pbds::gp_hash_table&lt; Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc &gt;</code> Class Template Reference	872
4.330.1 Detailed Description	873
4.330.2 Constructor & Destructor Documentation	873
4.331 <code>__gnu_pbds::gp_hash_tag</code> Struct Reference	877
4.331.1 Detailed Description	877
4.332 <code>__gnu_pbds::hash_exponential_size_policy&lt; Size_Type &gt;</code> Class Template Reference	877
4.332.1 Detailed Description	878
4.332.2 Constructor & Destructor Documentation	878
4.333 <code>__gnu_pbds::hash_load_check_resize_trigger&lt; External_Load_Access, Size_Type &gt;</code> Class Template Reference	878
4.333.1 Detailed Description	879
4.333.2 Member Enumeration Documentation	879
4.333.3 Constructor & Destructor Documentation	879
4.333.4 Member Function Documentation	880

4.334 <a href="#">__gnu_pbds::hash_prime_size_policy</a> Class Reference . . . . .	880
4.334.1 Detailed Description . . . . .	881
4.334.2 Member Typedef Documentation . . . . .	881
4.334.3 Constructor & Destructor Documentation . . . . .	881
4.335 <a href="#">__gnu_pbds::hash_standard_resize_policy&lt; Size_Policy, Trigger_Policy, External_Size_Access, Size_↵_Type &gt;</a> Class Template Reference . . . . .	881
4.335.1 Detailed Description . . . . .	882
4.335.2 Constructor & Destructor Documentation . . . . .	882
4.335.3 Member Function Documentation . . . . .	883
4.336 <a href="#">__gnu_pbds::insert_error</a> Struct Reference . . . . .	884
4.336.1 Detailed Description . . . . .	884
4.337 <a href="#">__gnu_pbds::join_error</a> Struct Reference . . . . .	885
4.337.1 Detailed Description . . . . .	885
4.338 <a href="#">__gnu_pbds::linear_probe_fn&lt; Size_Type &gt;</a> Class Template Reference . . . . .	885
4.338.1 Detailed Description . . . . .	885
4.338.2 Member Function Documentation . . . . .	886
4.339 <a href="#">__gnu_pbds::list_update&lt; Key, Mapped, Eq_Fn, Update_Policy, _Alloc &gt;</a> Class Template Reference . . . . .	886
4.339.1 Detailed Description . . . . .	886
4.339.2 Constructor & Destructor Documentation . . . . .	886
4.340 <a href="#">__gnu_pbds::list_update_tag</a> Struct Reference . . . . .	887
4.340.1 Detailed Description . . . . .	887
4.341 <a href="#">__gnu_pbds::lu_counter_policy&lt; Max_Count, _Alloc &gt;</a> Class Template Reference . . . . .	888
4.341.1 Detailed Description . . . . .	888
4.341.2 Member Typedef Documentation . . . . .	889
4.341.3 Member Enumeration Documentation . . . . .	889
4.341.4 Member Function Documentation . . . . .	889
4.342 <a href="#">__gnu_pbds::lu_move_to_front_policy&lt; _Alloc &gt;</a> Class Template Reference . . . . .	889
4.342.1 Detailed Description . . . . .	890
4.342.2 Member Typedef Documentation . . . . .	890
4.342.3 Member Function Documentation . . . . .	890
4.343 <a href="#">__gnu_pbds::null_node_update&lt; _Tp1, _Tp2, _Tp3, _Tp4 &gt;</a> Struct Template Reference . . . . .	891
4.343.1 Detailed Description . . . . .	891
4.344 <a href="#">__gnu_pbds::null_type</a> Struct Reference . . . . .	891
4.344.1 Detailed Description . . . . .	892
4.345 <a href="#">__gnu_pbds::ov_tree_tag</a> Struct Reference . . . . .	892
4.345.1 Detailed Description . . . . .	892
4.346 <a href="#">__gnu_pbds::pairing_heap_tag</a> Struct Reference . . . . .	893

4.346.1 Detailed Description . . . . .	893
4.347 __gnu_pbds::pat_trie_tag Struct Reference . . . . .	894
4.347.1 Detailed Description . . . . .	894
4.348 __gnu_pbds::point_invalidation_guarantee Struct Reference . . . . .	895
4.348.1 Detailed Description . . . . .	895
4.349 __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc > Class Template Reference . . . . .	895
4.349.1 Detailed Description . . . . .	896
4.350 __gnu_pbds::priority_queue_tag Struct Reference . . . . .	897
4.350.1 Detailed Description . . . . .	897
4.351 __gnu_pbds::quadratic_probe_fn< Size_Type > Class Template Reference . . . . .	897
4.351.1 Detailed Description . . . . .	897
4.351.2 Member Function Documentation . . . . .	898
4.352 __gnu_pbds::range_invalidation_guarantee Struct Reference . . . . .	898
4.352.1 Detailed Description . . . . .	898
4.353 __gnu_pbds::rb_tree_tag Struct Reference . . . . .	899
4.353.1 Detailed Description . . . . .	899
4.354 __gnu_pbds::rc_binomial_heap_tag Struct Reference . . . . .	900
4.354.1 Detailed Description . . . . .	900
4.355 __gnu_pbds::resize_error Struct Reference . . . . .	901
4.355.1 Detailed Description . . . . .	901
4.356 __gnu_pbds::sample_probe_fn Class Reference . . . . .	901
4.356.1 Detailed Description . . . . .	901
4.356.2 Constructor & Destructor Documentation . . . . .	902
4.356.3 Member Function Documentation . . . . .	902
4.357 __gnu_pbds::sample_range_hashing Class Reference . . . . .	902
4.357.1 Detailed Description . . . . .	902
4.357.2 Member Typedef Documentation . . . . .	903
4.357.3 Constructor & Destructor Documentation . . . . .	903
4.357.4 Member Function Documentation . . . . .	903
4.358 __gnu_pbds::sample_ranged_hash_fn Class Reference . . . . .	903
4.358.1 Detailed Description . . . . .	904
4.358.2 Constructor & Destructor Documentation . . . . .	904
4.358.3 Member Function Documentation . . . . .	904
4.359 __gnu_pbds::sample_ranged_probe_fn Class Reference . . . . .	904
4.359.1 Detailed Description . . . . .	905
4.360 __gnu_pbds::sample_resize_policy Class Reference . . . . .	905
4.360.1 Detailed Description . . . . .	905

4.360.2 Member Typedef Documentation . . . . .	905
4.360.3 Constructor & Destructor Documentation . . . . .	906
4.360.4 Member Function Documentation . . . . .	906
4.361 <code>__gnu_pbds::sample_resize_trigger</code> Class Reference . . . . .	907
4.361.1 Detailed Description . . . . .	908
4.361.2 Member Typedef Documentation . . . . .	908
4.361.3 Constructor & Destructor Documentation . . . . .	908
4.361.4 Member Function Documentation . . . . .	908
4.362 <code>__gnu_pbds::sample_size_policy</code> Class Reference . . . . .	909
4.362.1 Detailed Description . . . . .	910
4.362.2 Member Typedef Documentation . . . . .	910
4.362.3 Constructor & Destructor Documentation . . . . .	910
4.362.4 Member Function Documentation . . . . .	910
4.363 <code>__gnu_pbds::sample_tree_node_update&lt; Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc &gt;</code> Class Template Reference . . . . .	910
4.363.1 Detailed Description . . . . .	911
4.364 <code>__gnu_pbds::sample_trie_access_traits</code> Struct Reference . . . . .	911
4.364.1 Detailed Description . . . . .	911
4.364.2 Member Typedef Documentation . . . . .	911
4.364.3 Member Function Documentation . . . . .	911
4.365 <code>__gnu_pbds::sample_trie_node_update&lt; Node_Cltr, Node_Itr, _ATraits, _Alloc &gt;</code> Class Template Reference . . . . .	912
4.365.1 Detailed Description . . . . .	912
4.365.2 Constructor & Destructor Documentation . . . . .	912
4.365.3 Member Function Documentation . . . . .	912
4.366 <code>__gnu_pbds::sample_update_policy</code> Struct Reference . . . . .	913
4.366.1 Detailed Description . . . . .	913
4.366.2 Member Typedef Documentation . . . . .	913
4.366.3 Constructor & Destructor Documentation . . . . .	913
4.366.4 Member Function Documentation . . . . .	913
4.367 <code>__gnu_pbds::sequence_tag</code> Struct Reference . . . . .	914
4.367.1 Detailed Description . . . . .	914
4.368 <code>__gnu_pbds::splay_tree_tag</code> Struct Reference . . . . .	915
4.368.1 Detailed Description . . . . .	915
4.369 <code>__gnu_pbds::string_tag</code> Struct Reference . . . . .	916
4.369.1 Detailed Description . . . . .	916
4.370 <code>__gnu_pbds::thin_heap_tag</code> Struct Reference . . . . .	917

4.370.1 Detailed Description . . . . .	917
4.371 <code>__gnu_pbds::tree&lt; Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc &gt;</code> Class Template Reference . . . . .	917
4.371.1 Detailed Description . . . . .	918
4.371.2 Member Typedef Documentation . . . . .	918
4.371.3 Constructor & Destructor Documentation . . . . .	918
4.372 <code>__gnu_pbds::tree_order_statistics_node_update&lt; Node_Cltr, Node_Itr, Cmp_Fn, _Alloc &gt;</code> Class Template Reference . . . . .	919
4.372.1 Detailed Description . . . . .	920
4.372.2 Member Function Documentation . . . . .	921
4.373 <code>__gnu_pbds::tree_tag</code> Struct Reference . . . . .	922
4.373.1 Detailed Description . . . . .	922
4.374 <code>__gnu_pbds::trie&lt; Key, Mapped, _ATraits, Tag, Node_Update, _Alloc &gt;</code> Class Template Reference . . . . .	922
4.374.1 Detailed Description . . . . .	923
4.374.2 Member Typedef Documentation . . . . .	923
4.374.3 Constructor & Destructor Documentation . . . . .	923
4.375 <code>__gnu_pbds::trie_order_statistics_node_update&lt; Node_Cltr, Node_Itr, _ATraits, _Alloc &gt;</code> Class Template Reference . . . . .	925
4.375.1 Detailed Description . . . . .	926
4.375.2 Member Function Documentation . . . . .	926
4.376 <code>__gnu_pbds::trie_prefix_search_node_update&lt; Node_Cltr, Node_Itr, _ATraits, _Alloc &gt;</code> Class Template Reference . . . . .	928
4.376.1 Detailed Description . . . . .	929
4.376.2 Member Typedef Documentation . . . . .	929
4.376.3 Member Function Documentation . . . . .	930
4.377 <code>__gnu_pbds::trie_string_access_traits&lt; String, Min_E_Val, Max_E_Val, Reverse, _Alloc &gt;</code> Struct Template Reference . . . . .	931
4.377.1 Detailed Description . . . . .	931
4.377.2 Member Typedef Documentation . . . . .	932
4.377.3 Member Function Documentation . . . . .	932
4.378 <code>__gnu_pbds::trie_tag</code> Struct Reference . . . . .	933
4.378.1 Detailed Description . . . . .	933
4.379 <code>__gnu_pbds::trivial_iterator_tag</code> Struct Reference . . . . .	933
4.379.1 Detailed Description . . . . .	933
4.380 <code>__gnu_profile::__container_size_info</code> Class Reference . . . . .	934
4.380.1 Detailed Description . . . . .	935
4.381 <code>__gnu_profile::__container_size_stack_info</code> Class Reference . . . . .	935
4.381.1 Detailed Description . . . . .	936
4.382 <code>__gnu_profile::__hashfunc_info</code> Class Reference . . . . .	936



4.382.1 Detailed Description	937
4.383 <code>__gnu_profile::__hashfunc_stack_info</code> Class Reference	937
4.383.1 Detailed Description	938
4.384 <code>__gnu_profile::__list2vector_info</code> Class Reference	938
4.384.1 Detailed Description	939
4.385 <code>__gnu_profile::__map2umap_info</code> Class Reference	939
4.385.1 Detailed Description	940
4.386 <code>__gnu_profile::__map2umap_stack_info</code> Class Reference	940
4.386.1 Detailed Description	941
4.387 <code>__gnu_profile::__object_info_base</code> Class Reference	941
4.387.1 Detailed Description	942
4.388 <code>__gnu_profile::__reentrance_guard</code> Struct Reference	942
4.388.1 Detailed Description	942
4.389 <code>__gnu_profile::__stack_hash</code> Class Reference	942
4.389.1 Detailed Description	942
4.390 <code>__gnu_profile::__stack_info_base&lt; __object_info &gt;</code> Class Template Reference	942
4.390.1 Detailed Description	943
4.391 <code>__gnu_profile::__trace_base&lt; __object_info, __stack_info &gt;</code> Class Template Reference	943
4.391.1 Detailed Description	943
4.392 <code>__gnu_profile::__trace_container_size</code> Class Reference	943
4.392.1 Detailed Description	944
4.393 <code>__gnu_profile::__trace_hash_func</code> Class Reference	944
4.393.1 Detailed Description	945
4.394 <code>__gnu_profile::__trace_hashtable_size</code> Class Reference	945
4.394.1 Detailed Description	946
4.395 <code>__gnu_profile::__trace_map2umap</code> Class Reference	946
4.395.1 Detailed Description	946
4.396 <code>__gnu_profile::__trace_vector_size</code> Class Reference	947
4.396.1 Detailed Description	947
4.397 <code>__gnu_profile::__trace_vector_to_list</code> Class Reference	948
4.397.1 Detailed Description	948
4.398 <code>__gnu_profile::__vector2list_info</code> Class Reference	949
4.398.1 Detailed Description	950
4.399 <code>__gnu_profile::__vector2list_stack_info</code> Class Reference	950
4.399.1 Detailed Description	951
4.400 <code>__gnu_profile::__warning_data</code> Struct Reference	951
4.400.1 Detailed Description	951

4.401	<code>const_iterator_</code> Class Reference	952
4.401.1	Detailed Description	953
4.401.2	Member Typedef Documentation	953
4.401.3	Constructor & Destructor Documentation	954
4.401.4	Member Function Documentation	954
4.401.5	Member Data Documentation	955
4.402	<code>iterator_</code> Class Reference	955
4.402.1	Detailed Description	956
4.402.2	Member Typedef Documentation	957
4.402.3	Constructor & Destructor Documentation	957
4.402.4	Member Function Documentation	958
4.402.5	Member Data Documentation	959
4.403	<code>point_const_iterator_</code> Class Reference	959
4.403.1	Detailed Description	960
4.403.2	Member Typedef Documentation	960
4.403.3	Constructor & Destructor Documentation	961
4.403.4	Member Function Documentation	961
4.404	<code>point_iterator_</code> Class Reference	962
4.404.1	Detailed Description	963
4.404.2	Member Typedef Documentation	963
4.404.3	Constructor & Destructor Documentation	963
4.404.4	Member Function Documentation	964
4.405	<code>std::__atomic_base&lt;_ITp&gt;</code> Struct Template Reference	964
4.405.1	Detailed Description	966
4.406	<code>std::__atomic_base&lt;_PTp*&gt;</code> Struct Template Reference	966
4.406.1	Detailed Description	967
4.407	<code>std::__atomic_flag_base</code> Struct Reference	967
4.407.1	Detailed Description	967
4.408	<code>std::__codecvt_abstract_base&lt;_InternT, _ExternT, _StateT&gt;</code> Class Template Reference	968
4.408.1	Detailed Description	969
4.408.2	Member Function Documentation	969
4.409	<code>std::__ctype_abstract_base&lt;_CharT&gt;</code> Class Template Reference	972
4.409.1	Detailed Description	973
4.409.2	Member Typedef Documentation	974
4.409.3	Member Function Documentation	974
4.410	<code>std::__debug::map&lt;_Key, _Tp, _Compare, _Allocator&gt;</code> Class Template Reference	983
4.410.1	Detailed Description	985

4.410.2 Member Function Documentation	985
4.410.3 Member Data Documentation	987
4.411std::__debug::multimap<_Key, _Tp, _Compare, _Allocator> Class Template Reference	987
4.411.1 Detailed Description	989
4.411.2 Member Function Documentation	990
4.411.3 Member Data Documentation	991
4.412std::__debug::multiset<_Key, _Compare, _Allocator> Class Template Reference	992
4.412.1 Detailed Description	994
4.412.2 Member Function Documentation	994
4.412.3 Member Data Documentation	995
4.413std::__debug::set<_Key, _Compare, _Allocator> Class Template Reference	996
4.413.1 Detailed Description	998
4.413.2 Member Function Documentation	998
4.413.3 Member Data Documentation	999
4.414std::__detail::_Automaton Class Reference	1000
4.414.1 Detailed Description	1000
4.415std::__detail::_Before_begin<_NodeAlloc> Struct Template Reference	1000
4.415.1 Detailed Description	1001
4.416std::__detail::_CharMatcher<_InIterT, _TraitsT> Struct Template Reference	1001
4.416.1 Detailed Description	1001
4.417std::__detail::_Compiler<_InIter, _TraitsT> Class Template Reference	1001
4.417.1 Detailed Description	1002
4.418std::__detail::_Default_ranged_hash Struct Reference	1002
4.418.1 Detailed Description	1002
4.419std::__detail::_EndTagger<_FwdIterT, _TraitsT> Struct Template Reference	1002
4.419.1 Detailed Description	1002
4.420std::__detail::_Equal_helper<_Key, _Value, _ExtractKey, _Equal, _HashCodeType, __cache_hash↵code> Struct Template Reference	1003
4.420.1 Detailed Description	1003
4.421std::__detail::_Equal_helper<_Key, _Value, _ExtractKey, _Equal, _HashCodeType, false> Struct Template Reference	1003
4.421.1 Detailed Description	1003
4.422std::__detail::_Equal_helper<_Key, _Value, _ExtractKey, _Equal, _HashCodeType, true> Struct Template Reference	1003
4.422.1 Detailed Description	1003
4.423std::__detail::_Equality<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys> Struct Template Reference	1004
4.423.1 Detailed Description	1004

4.424std::__detail::Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false > Struct Template Reference . . . . .	1005
4.424.1 Detailed Description . . . . .	1005
4.425std::__detail::Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true > Struct Template Reference . . . . .	1006
4.425.1 Detailed Description . . . . .	1006
4.426std::__detail::Equality_base Struct Reference . . . . .	1006
4.426.1 Detailed Description . . . . .	1006
4.427std::__detail::Grep_matcher Class Reference . . . . .	1007
4.427.1 Detailed Description . . . . .	1007
4.428std::__detail::Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code > Struct Template Reference . . . . .	1007
4.428.1 Detailed Description . . . . .	1007
4.429std::__detail::Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false > Struct Template Reference . . . . .	1008
4.429.1 Detailed Description . . . . .	1009
4.430std::__detail::Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true > Struct Template Reference . . . . .	1009
4.430.1 Detailed Description . . . . .	1010
4.431std::__detail::Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false > Struct Template Reference . . . . .	1010
4.431.1 Detailed Description . . . . .	1011
4.432std::__detail::Hash_node< _Value, _Cache_hash_code > Struct Template Reference . . . . .	1011
4.432.1 Detailed Description . . . . .	1011
4.433std::__detail::Hash_node< _Value, false > Struct Template Reference . . . . .	1012
4.433.1 Detailed Description . . . . .	1012
4.434std::__detail::Hash_node< _Value, true > Struct Template Reference . . . . .	1013
4.434.1 Detailed Description . . . . .	1013
4.435std::__detail::Hash_node_base Struct Reference . . . . .	1014
4.435.1 Detailed Description . . . . .	1014
4.436std::__detail::Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits > Struct Template Reference . . . . .	1015
4.436.1 Detailed Description . . . . .	1016
4.437std::__detail::Hashtable_ebo_helper< _Nm, _Tp, __use_ebo > Struct Template Reference . . . . .	1016
4.437.1 Detailed Description . . . . .	1016
4.438std::__detail::Hashtable_ebo_helper< _Nm, _Tp, false > Struct Template Reference . . . . .	1016
4.438.1 Detailed Description . . . . .	1017
4.439std::__detail::Hashtable_ebo_helper< _Nm, _Tp, true > Struct Template Reference . . . . .	1017
4.439.1 Detailed Description . . . . .	1017

4.440std::__detail::Hashtable_traits< _Cache_hash_code, _Constant_iterators, _Unique_keys > Struct Template Reference	1017
4.440.1 Detailed Description	1018
4.441std::__detail::Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Constant_iterators, _Unique_keys > Struct Template Reference	1018
4.441.1 Detailed Description	1018
4.442std::__detail::Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false, _Unique_keys > Struct Template Reference	1019
4.442.1 Detailed Description	1020
4.443std::__detail::Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, false > Struct Template Reference	1020
4.443.1 Detailed Description	1021
4.444std::__detail::Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, true > Struct Template Reference	1021
4.444.1 Detailed Description	1022
4.445std::__detail::Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits > Struct Template Reference	1023
4.445.1 Detailed Description	1024
4.446std::__detail::List_node_base Struct Reference	1024
4.446.1 Detailed Description	1025
4.447std::__detail::Local_const_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache > Struct Template Reference	1025
4.447.1 Detailed Description	1025
4.448std::__detail::Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache > Struct Template Reference	1026
4.448.1 Detailed Description	1026
4.449std::__detail::Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code > Struct Template Reference	1027
4.449.1 Detailed Description	1027
4.450std::__detail::Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false > Struct Template Reference	1027
4.450.1 Detailed Description	1028
4.451std::__detail::Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true > Struct Template Reference	1029
4.451.1 Detailed Description	1029
4.452std::__detail::Map_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys > Struct Template Reference	1030
4.452.1 Detailed Description	1030
4.453std::__detail::Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false > Struct Template Reference	1030
4.453.1 Detailed Description	1030

4.454std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true > Struct Template Reference	1031
4.454.1 Detailed Description	1031
4.455std::__detail::_Mod_range_hashing Struct Reference	1031
4.455.1 Detailed Description	1032
4.456std::__detail::_Nfa Class Reference	1032
4.456.1 Detailed Description	1035
4.456.2 Member Function Documentation	1035
4.457std::__detail::_Node_const_iterator< _Value, __constant_iterators, __cache > Struct Template Reference	1046
4.457.1 Detailed Description	1046
4.458std::__detail::_Node_iterator< _Value, __constant_iterators, __cache > Struct Template Reference	1047
4.458.1 Detailed Description	1048
4.459std::__detail::_Node_iterator_base< _Value, _Cache_hash_code > Struct Template Reference	1048
4.459.1 Detailed Description	1048
4.460std::__detail::_PatternCursor Struct Reference	1049
4.460.1 Detailed Description	1049
4.461std::__detail::_Prime_rehash_policy Struct Reference	1049
4.461.1 Detailed Description	1050
4.462std::__detail::_RangeMatcher< _InIterT, _TraitsT > Struct Template Reference	1050
4.462.1 Detailed Description	1050
4.463std::__detail::_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits > Struct Template Reference	1051
4.463.1 Detailed Description	1051
4.464std::__detail::_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _Prime_rehash_policy, _Traits > Struct Template Reference	1051
4.464.1 Detailed Description	1052
4.465std::__detail::_Results Struct Reference	1052
4.465.1 Detailed Description	1052
4.466std::__detail::_Scanner< _InputIterator > Class Template Reference	1053
4.466.1 Detailed Description	1054
4.467std::__detail::_Scanner_base Struct Reference	1054
4.467.1 Detailed Description	1054
4.468std::__detail::_SpecializedCursor< _FwdIterT > Class Template Reference	1055
4.468.1 Detailed Description	1055
4.469std::__detail::_SpecializedResults< _FwdIterT, _Alloc > Class Template Reference	1056
4.469.1 Detailed Description	1056
4.470std::__detail::_StartTagger< _FwdIterT, _TraitsT > Struct Template Reference	1056
4.470.1 Detailed Description	1057

4.471	<a href="#">std::__detail::_State Struct Reference</a>	1057
4.471.1	Detailed Description	1057
4.472	<a href="#">std::__detail::_StateSeq Class Reference</a>	1058
4.472.1	Detailed Description	1058
4.473	<a href="#">std::__exception_ptr::exception_ptr Class Reference</a>	1058
4.473.1	Detailed Description	1058
4.474	<a href="#">std::__has_iterator_category_helper&lt; _Tp &gt; Class Template Reference</a>	1059
4.474.1	Detailed Description	1059
4.475	<a href="#">std::__parallel::_CRandNumber&lt; _MustBeInt &gt; Struct Template Reference</a>	1059
4.475.1	Detailed Description	1059
4.476	<a href="#">std::__profile::map&lt; _Key, _Tp, _Compare, _Allocator &gt; Class Template Reference</a>	1059
4.476.1	Detailed Description	1061
4.477	<a href="#">std::__profile::multimap&lt; _Key, _Tp, _Compare, _Allocator &gt; Class Template Reference</a>	1061
4.477.1	Detailed Description	1063
4.478	<a href="#">std::__profile::multiset&lt; _Key, _Compare, _Allocator &gt; Class Template Reference</a>	1063
4.478.1	Detailed Description	1064
4.479	<a href="#">std::__profile::set&lt; _Key, _Compare, _Allocator &gt; Class Template Reference</a>	1065
4.479.1	Detailed Description	1066
4.480	<a href="#">std::_Deque_base&lt; _Tp, _Alloc &gt; Class Template Reference</a>	1066
4.480.1	Detailed Description	1067
4.480.2	Member Function Documentation	1068
4.481	<a href="#">std::_Deque_iterator&lt; _Tp, _Ref, _Ptr &gt; Struct Template Reference</a>	1069
4.481.1	Detailed Description	1070
4.481.2	Member Function Documentation	1070
4.482	<a href="#">std::_Fwd_list_base&lt; _Tp, _Alloc &gt; Struct Template Reference</a>	1070
4.482.1	Detailed Description	1071
4.483	<a href="#">std::_Fwd_list_const_iterator&lt; _Tp &gt; Struct Template Reference</a>	1072
4.483.1	Detailed Description	1072
4.484	<a href="#">std::_Fwd_list_iterator&lt; _Tp &gt; Struct Template Reference</a>	1072
4.484.1	Detailed Description	1073
4.485	<a href="#">std::_Fwd_list_node&lt; _Tp &gt; Struct Template Reference</a>	1073
4.485.1	Detailed Description	1074
4.486	<a href="#">std::_Fwd_list_node_base Struct Reference</a>	1074
4.486.1	Detailed Description	1075
4.487	<a href="#">std::_Hashtable&lt; _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits &gt; Class Template Reference</a>	1075
4.487.1	Detailed Description	1078

4.488	<a href="#">std::_List_base&lt;_Tp, _Alloc &gt; Class Template Reference</a>	1080
4.488.1	<a href="#">Detailed Description</a>	1081
4.489	<a href="#">std::_List_const_iterator&lt;_Tp &gt; Struct Template Reference</a>	1081
4.489.1	<a href="#">Detailed Description</a>	1081
4.490	<a href="#">std::_List_iterator&lt;_Tp &gt; Struct Template Reference</a>	1082
4.490.1	<a href="#">Detailed Description</a>	1082
4.491	<a href="#">std::_List_node&lt;_Tp &gt; Struct Template Reference</a>	1083
4.491.1	<a href="#">Detailed Description</a>	1083
4.491.2	<a href="#">Member Data Documentation</a>	1083
4.492	<a href="#">std::_Temporary_buffer&lt;_ForwardIterator, _Tp &gt; Class Template Reference</a>	1084
4.492.1	<a href="#">Detailed Description</a>	1084
4.492.2	<a href="#">Constructor &amp; Destructor Documentation</a>	1084
4.492.3	<a href="#">Member Function Documentation</a>	1085
4.493	<a href="#">std::_Vector_base&lt;_Tp, _Alloc &gt; Struct Template Reference</a>	1086
4.493.1	<a href="#">Detailed Description</a>	1087
4.494	<a href="#">std::allocator&lt;_Tp &gt; Class Template Reference</a>	1087
4.494.1	<a href="#">Detailed Description</a>	1088
4.495	<a href="#">std::allocator&lt;void &gt; Class Template Reference</a>	1088
4.495.1	<a href="#">Detailed Description</a>	1088
4.496	<a href="#">std::allocator_arg_t Struct Reference</a>	1088
4.496.1	<a href="#">Detailed Description</a>	1088
4.497	<a href="#">std::allocator_traits&lt;_Alloc &gt; Struct Template Reference</a>	1089
4.497.1	<a href="#">Detailed Description</a>	1090
4.497.2	<a href="#">Member Typedef Documentation</a>	1090
4.497.3	<a href="#">Member Function Documentation</a>	1091
4.498	<a href="#">std::atomic_flag Struct Reference</a>	1094
4.498.1	<a href="#">Detailed Description</a>	1095
4.499	<a href="#">std::auto_ptr&lt;_Tp &gt; Class Template Reference</a>	1095
4.499.1	<a href="#">Detailed Description</a>	1096
4.499.2	<a href="#">Member Typedef Documentation</a>	1096
4.499.3	<a href="#">Constructor &amp; Destructor Documentation</a>	1096
4.499.4	<a href="#">Member Function Documentation</a>	1097
4.500	<a href="#">std::auto_ptr_ref&lt;_Tp1 &gt; Struct Template Reference</a>	1099
4.500.1	<a href="#">Detailed Description</a>	1099
4.501	<a href="#">std::back_insert_iterator&lt;_Container &gt; Class Template Reference</a>	1100
4.501.1	<a href="#">Detailed Description</a>	1100
4.501.2	<a href="#">Member Typedef Documentation</a>	1101



4.501.3 Constructor & Destructor Documentation . . . . .	1101
4.501.4 Member Function Documentation . . . . .	1101
4.502std::bad_weak_ptr Class Reference . . . . .	1102
4.502.1 Detailed Description . . . . .	1102
4.503std::basic_ios< _CharT, _Traits > Class Template Reference . . . . .	1103
4.503.1 Detailed Description . . . . .	1106
4.503.2 Member Typedef Documentation . . . . .	1106
4.503.3 Member Enumeration Documentation . . . . .	1109
4.503.4 Constructor & Destructor Documentation . . . . .	1109
4.503.5 Member Function Documentation . . . . .	1109
4.503.6 Member Data Documentation . . . . .	1120
4.504std::basic_regex< _Ch_type, _Rx_traits > Class Template Reference . . . . .	1123
4.504.1 Detailed Description . . . . .	1125
4.504.2 Constructor & Destructor Documentation . . . . .	1125
4.504.3 Member Function Documentation . . . . .	1127
4.505std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference . . . . .	1132
4.505.1 Detailed Description . . . . .	1135
4.505.2 Constructor & Destructor Documentation . . . . .	1136
4.505.3 Member Function Documentation . . . . .	1138
4.505.4 Member Data Documentation . . . . .	1176
4.506std::bernoulli_distribution Class Reference . . . . .	1176
4.506.1 Detailed Description . . . . .	1177
4.506.2 Member Typedef Documentation . . . . .	1177
4.506.3 Constructor & Destructor Documentation . . . . .	1177
4.506.4 Member Function Documentation . . . . .	1178
4.506.5 Friends And Related Function Documentation . . . . .	1178
4.507std::bernoulli_distribution::param_type Struct Reference . . . . .	1179
4.507.1 Detailed Description . . . . .	1179
4.508std::bidirectional_iterator_tag Struct Reference . . . . .	1180
4.508.1 Detailed Description . . . . .	1180
4.509std::binary_function< _Arg1, _Arg2, _Result > Struct Template Reference . . . . .	1181
4.509.1 Detailed Description . . . . .	1181
4.509.2 Member Typedef Documentation . . . . .	1181
4.510std::binary_negate< _Predicate > Class Template Reference . . . . .	1182
4.510.1 Detailed Description . . . . .	1182
4.510.2 Member Typedef Documentation . . . . .	1183
4.511std::binder1st< _Operation > Class Template Reference . . . . .	1183

4.511.1 Detailed Description . . . . .	1184
4.511.2 Member Typedef Documentation . . . . .	1184
4.512std::binder2nd< _Operation > Class Template Reference . . . . .	1185
4.512.1 Detailed Description . . . . .	1185
4.512.2 Member Typedef Documentation . . . . .	1185
4.513std::binomial_distribution< _IntType > Class Template Reference . . . . .	1186
4.513.1 Detailed Description . . . . .	1187
4.513.2 Member Typedef Documentation . . . . .	1187
4.513.3 Member Function Documentation . . . . .	1187
4.513.4 Friends And Related Function Documentation . . . . .	1189
4.514std::binomial_distribution< _IntType >::param_type Struct Reference . . . . .	1190
4.514.1 Detailed Description . . . . .	1190
4.515std::cauchy_distribution< _RealType > Class Template Reference . . . . .	1190
4.515.1 Detailed Description . . . . .	1191
4.515.2 Member Typedef Documentation . . . . .	1191
4.515.3 Member Function Documentation . . . . .	1191
4.515.4 Friends And Related Function Documentation . . . . .	1192
4.516std::cauchy_distribution< _RealType >::param_type Struct Reference . . . . .	1192
4.516.1 Detailed Description . . . . .	1193
4.517std::char_traits< _CharT > Struct Template Reference . . . . .	1193
4.517.1 Detailed Description . . . . .	1194
4.518std::char_traits< __gnu_cxx::character< V, I, S > > Struct Template Reference . . . . .	1194
4.518.1 Detailed Description . . . . .	1195
4.519std::char_traits< char > Struct Template Reference . . . . .	1195
4.519.1 Detailed Description . . . . .	1196
4.520std::char_traits< wchar_t > Struct Template Reference . . . . .	1196
4.520.1 Detailed Description . . . . .	1196
4.521std::chi_squared_distribution< _RealType > Class Template Reference . . . . .	1197
4.521.1 Detailed Description . . . . .	1198
4.521.2 Member Typedef Documentation . . . . .	1198
4.521.3 Member Function Documentation . . . . .	1198
4.521.4 Friends And Related Function Documentation . . . . .	1199
4.522std::chi_squared_distribution< _RealType >::param_type Struct Reference . . . . .	1200
4.522.1 Detailed Description . . . . .	1200
4.523std::codecvt< _InternT, _ExternT, _StateT > Class Template Reference . . . . .	1201
4.523.1 Detailed Description . . . . .	1202
4.523.2 Member Function Documentation . . . . .	1202

4.524	<a href="#">std::codecvt&lt; _InternT, _ExternT, encoding_state &gt; Class Template Reference</a>	1205
4.524.1	Detailed Description	1206
4.524.2	Member Function Documentation	1206
4.525	<a href="#">std::codecvt&lt; char, char, mbstate_t &gt; Class Template Reference</a>	1209
4.525.1	Detailed Description	1210
4.525.2	Member Function Documentation	1210
4.526	<a href="#">std::codecvt&lt; wchar_t, char, mbstate_t &gt; Class Template Reference</a>	1213
4.526.1	Detailed Description	1214
4.526.2	Member Function Documentation	1214
4.527	<a href="#">std::codecvt_base Class Reference</a>	1217
4.527.1	Detailed Description	1217
4.528	<a href="#">std::codecvt_byname&lt; _InternT, _ExternT, _StateT &gt; Class Template Reference</a>	1218
4.528.1	Detailed Description	1219
4.528.2	Member Function Documentation	1219
4.529	<a href="#">std::collate&lt; _CharT &gt; Class Template Reference</a>	1222
4.529.1	Detailed Description	1223
4.529.2	Member Typedef Documentation	1223
4.529.3	Constructor & Destructor Documentation	1223
4.529.4	Member Function Documentation	1225
4.529.5	Member Data Documentation	1227
4.530	<a href="#">std::collate_byname&lt; _CharT &gt; Class Template Reference</a>	1228
4.530.1	Detailed Description	1229
4.530.2	Member Typedef Documentation	1229
4.530.3	Member Function Documentation	1229
4.530.4	Member Data Documentation	1231
4.531	<a href="#">std::const_mem_fun1_ref_t&lt; _Ret, _Tp, _Arg &gt; Class Template Reference</a>	1232
4.531.1	Detailed Description	1232
4.531.2	Member Typedef Documentation	1232
4.532	<a href="#">std::const_mem_fun1_t&lt; _Ret, _Tp, _Arg &gt; Class Template Reference</a>	1233
4.532.1	Detailed Description	1233
4.532.2	Member Typedef Documentation	1234
4.533	<a href="#">std::const_mem_fun_ref_t&lt; _Ret, _Tp &gt; Class Template Reference</a>	1234
4.533.1	Detailed Description	1235
4.533.2	Member Typedef Documentation	1235
4.534	<a href="#">std::const_mem_fun_t&lt; _Ret, _Tp &gt; Class Template Reference</a>	1235
4.534.1	Detailed Description	1236
4.534.2	Member Typedef Documentation	1236

4.535	<code>std::ctype&lt; _CharT &gt;</code> Class Template Reference	1237
4.535.1	Detailed Description	1238
4.535.2	Member Function Documentation	1239
4.535.3	Member Data Documentation	1248
4.536	<code>std::ctype&lt; char &gt;</code> Class Template Reference	1249
4.536.1	Detailed Description	1251
4.536.2	Member Typedef Documentation	1251
4.536.3	Constructor & Destructor Documentation	1251
4.536.4	Member Function Documentation	1252
4.536.5	Member Data Documentation	1260
4.537	<code>std::ctype&lt; wchar_t &gt;</code> Class Template Reference	1261
4.537.1	Detailed Description	1263
4.537.2	Member Typedef Documentation	1263
4.537.3	Constructor & Destructor Documentation	1263
4.537.4	Member Function Documentation	1263
4.537.5	Member Data Documentation	1275
4.538	<code>std::ctype_base</code> Struct Reference	1276
4.538.1	Detailed Description	1276
4.539	<code>std::ctype_byname&lt; _CharT &gt;</code> Class Template Reference	1277
4.539.1	Detailed Description	1278
4.539.2	Member Function Documentation	1279
4.539.3	Member Data Documentation	1288
4.540	<code>std::ctype_byname&lt; char &gt;</code> Class Template Reference	1289
4.540.1	Detailed Description	1291
4.540.2	Member Typedef Documentation	1291
4.540.3	Member Function Documentation	1291
4.540.4	Member Data Documentation	1299
4.541	<code>std::default_delete&lt; _Tp &gt;</code> Struct Template Reference	1300
4.541.1	Detailed Description	1300
4.542	<code>std::default_delete&lt; _Tp[] &gt;</code> Struct Template Reference	1300
4.542.1	Detailed Description	1300
4.543	<code>std::deque&lt; _Tp, _Alloc &gt;</code> Class Template Reference	1301
4.543.1	Detailed Description	1304
4.543.2	Constructor & Destructor Documentation	1305
4.543.3	Member Function Documentation	1309
4.544	<code>std::discard_block_engine&lt; _RandomNumberEngine, __p, __r &gt;</code> Class Template Reference	1323
4.544.1	Detailed Description	1324

4.544.2 Member Typedef Documentation . . . . .	1324
4.544.3 Constructor & Destructor Documentation . . . . .	1324
4.544.4 Member Function Documentation . . . . .	1325
4.544.5 Friends And Related Function Documentation . . . . .	1326
4.545std::discrete_distribution<_IntType > Class Template Reference . . . . .	1327
4.545.1 Detailed Description . . . . .	1328
4.545.2 Member Typedef Documentation . . . . .	1328
4.545.3 Member Function Documentation . . . . .	1329
4.545.4 Friends And Related Function Documentation . . . . .	1329
4.546std::discrete_distribution<_IntType >::param_type Struct Reference . . . . .	1331
4.546.1 Detailed Description . . . . .	1332
4.547std::divides<_Tp > Struct Template Reference . . . . .	1332
4.547.1 Detailed Description . . . . .	1332
4.547.2 Member Typedef Documentation . . . . .	1333
4.548std::enable_shared_from_this<_Tp > Class Template Reference . . . . .	1333
4.548.1 Detailed Description . . . . .	1333
4.549std::equal_to<_Tp > Struct Template Reference . . . . .	1334
4.549.1 Detailed Description . . . . .	1334
4.549.2 Member Typedef Documentation . . . . .	1334
4.550std::exponential_distribution<_RealType > Class Template Reference . . . . .	1335
4.550.1 Detailed Description . . . . .	1335
4.550.2 Member Typedef Documentation . . . . .	1336
4.550.3 Constructor & Destructor Documentation . . . . .	1336
4.550.4 Member Function Documentation . . . . .	1336
4.550.5 Friends And Related Function Documentation . . . . .	1337
4.551std::exponential_distribution<_RealType >::param_type Struct Reference . . . . .	1337
4.551.1 Detailed Description . . . . .	1338
4.552std::extreme_value_distribution<_RealType > Class Template Reference . . . . .	1338
4.552.1 Detailed Description . . . . .	1339
4.552.2 Member Typedef Documentation . . . . .	1339
4.552.3 Member Function Documentation . . . . .	1339
4.552.4 Friends And Related Function Documentation . . . . .	1340
4.553std::extreme_value_distribution<_RealType >::param_type Struct Reference . . . . .	1340
4.553.1 Detailed Description . . . . .	1341
4.554std::fisher_f_distribution<_RealType > Class Template Reference . . . . .	1341
4.554.1 Detailed Description . . . . .	1342
4.554.2 Member Typedef Documentation . . . . .	1342

4.554.3 Member Function Documentation . . . . .	1342
4.554.4 Friends And Related Function Documentation . . . . .	1344
4.555std::fisher_f_distribution<_RealType>::param_type Struct Reference . . . . .	1345
4.555.1 Detailed Description . . . . .	1345
4.556std::forward_iterator_tag Struct Reference . . . . .	1346
4.556.1 Detailed Description . . . . .	1346
4.557std::forward_list<_Tp, _Alloc> Class Template Reference . . . . .	1347
4.557.1 Detailed Description . . . . .	1349
4.557.2 Constructor & Destructor Documentation . . . . .	1349
4.557.3 Member Function Documentation . . . . .	1351
4.558std::fpos<_StateT> Class Template Reference . . . . .	1363
4.558.1 Detailed Description . . . . .	1363
4.558.2 Constructor & Destructor Documentation . . . . .	1364
4.558.3 Member Function Documentation . . . . .	1364
4.559std::front_insert_iterator<_Container> Class Template Reference . . . . .	1365
4.559.1 Detailed Description . . . . .	1366
4.559.2 Member Typedef Documentation . . . . .	1366
4.559.3 Constructor & Destructor Documentation . . . . .	1366
4.559.4 Member Function Documentation . . . . .	1367
4.560std::gamma_distribution<_RealType> Class Template Reference . . . . .	1367
4.560.1 Detailed Description . . . . .	1368
4.560.2 Member Typedef Documentation . . . . .	1369
4.560.3 Constructor & Destructor Documentation . . . . .	1369
4.560.4 Member Function Documentation . . . . .	1369
4.560.5 Friends And Related Function Documentation . . . . .	1370
4.561std::gamma_distribution<_RealType>::param_type Struct Reference . . . . .	1371
4.561.1 Detailed Description . . . . .	1371
4.562std::geometric_distribution<_IntType> Class Template Reference . . . . .	1371
4.562.1 Detailed Description . . . . .	1372
4.562.2 Member Typedef Documentation . . . . .	1372
4.562.3 Member Function Documentation . . . . .	1372
4.562.4 Friends And Related Function Documentation . . . . .	1373
4.563std::geometric_distribution<_IntType>::param_type Struct Reference . . . . .	1374
4.563.1 Detailed Description . . . . .	1374
4.564std::greater<_Tp> Struct Template Reference . . . . .	1374
4.564.1 Detailed Description . . . . .	1375
4.564.2 Member Typedef Documentation . . . . .	1375

4.565std::greater_equal< _Tp > Struct Template Reference . . . . .	1375
4.565.1 Detailed Description . . . . .	1376
4.565.2 Member Typedef Documentation . . . . .	1376
4.566std::gslice Class Reference . . . . .	1376
4.566.1 Detailed Description . . . . .	1377
4.567std::gslice_array< _Tp > Class Template Reference . . . . .	1377
4.567.1 Detailed Description . . . . .	1378
4.568std::hash< _Tp > Struct Template Reference . . . . .	1378
4.568.1 Detailed Description . . . . .	1378
4.569std::hash< __gnu_cxx::__u16vstring > Struct Template Reference . . . . .	1378
4.569.1 Detailed Description . . . . .	1379
4.570std::hash< __gnu_cxx::__u32vstring > Struct Template Reference . . . . .	1379
4.570.1 Detailed Description . . . . .	1379
4.571std::hash< __gnu_cxx::__vstring > Struct Template Reference . . . . .	1379
4.571.1 Detailed Description . . . . .	1380
4.572std::hash< __gnu_cxx::__wvstring > Struct Template Reference . . . . .	1380
4.572.1 Detailed Description . . . . .	1380
4.573std::hash< __gnu_cxx::throw_value_limit > Struct Template Reference . . . . .	1381
4.573.1 Detailed Description . . . . .	1381
4.573.2 Member Typedef Documentation . . . . .	1381
4.574std::hash< __gnu_cxx::throw_value_random > Struct Template Reference . . . . .	1382
4.574.1 Detailed Description . . . . .	1382
4.574.2 Member Typedef Documentation . . . . .	1382
4.575std::hash< __shared_ptr< _Tp, _Lp > > Struct Template Reference . . . . .	1383
4.575.1 Detailed Description . . . . .	1383
4.576std::hash< _Tp * > Struct Template Reference . . . . .	1383
4.576.1 Detailed Description . . . . .	1384
4.577std::hash< bool > Struct Template Reference . . . . .	1384
4.577.1 Detailed Description . . . . .	1384
4.578std::hash< char > Struct Template Reference . . . . .	1384
4.578.1 Detailed Description . . . . .	1385
4.579std::hash< char16_t > Struct Template Reference . . . . .	1385
4.579.1 Detailed Description . . . . .	1385
4.580std::hash< char32_t > Struct Template Reference . . . . .	1385
4.580.1 Detailed Description . . . . .	1386
4.581std::hash< double > Struct Template Reference . . . . .	1386
4.581.1 Detailed Description . . . . .	1386

4.582std::hash< float > Struct Template Reference . . . . .	1386
4.582.1 Detailed Description . . . . .	1387
4.583std::hash< int > Struct Template Reference . . . . .	1387
4.583.1 Detailed Description . . . . .	1387
4.584std::hash< long > Struct Template Reference . . . . .	1387
4.584.1 Detailed Description . . . . .	1388
4.585std::hash< long double > Struct Template Reference . . . . .	1388
4.585.1 Detailed Description . . . . .	1388
4.586std::hash< long long > Struct Template Reference . . . . .	1388
4.586.1 Detailed Description . . . . .	1389
4.587std::hash< shared_ptr< _Tp > > Struct Template Reference . . . . .	1389
4.587.1 Detailed Description . . . . .	1389
4.588std::hash< short > Struct Template Reference . . . . .	1389
4.588.1 Detailed Description . . . . .	1390
4.589std::hash< signed char > Struct Template Reference . . . . .	1390
4.589.1 Detailed Description . . . . .	1390
4.590std::hash< string > Struct Template Reference . . . . .	1390
4.590.1 Detailed Description . . . . .	1391
4.591std::hash< u16string > Struct Template Reference . . . . .	1391
4.591.1 Detailed Description . . . . .	1391
4.592std::hash< u32string > Struct Template Reference . . . . .	1391
4.592.1 Detailed Description . . . . .	1392
4.593std::hash< unique_ptr< _Tp, _Dp > > Struct Template Reference . . . . .	1392
4.593.1 Detailed Description . . . . .	1392
4.594std::hash< unsigned char > Struct Template Reference . . . . .	1392
4.594.1 Detailed Description . . . . .	1393
4.595std::hash< unsigned int > Struct Template Reference . . . . .	1393
4.595.1 Detailed Description . . . . .	1393
4.596std::hash< unsigned long > Struct Template Reference . . . . .	1393
4.596.1 Detailed Description . . . . .	1394
4.597std::hash< unsigned long long > Struct Template Reference . . . . .	1394
4.597.1 Detailed Description . . . . .	1394
4.598std::hash< unsigned short > Struct Template Reference . . . . .	1394
4.598.1 Detailed Description . . . . .	1395
4.599std::hash< wchar_t > Struct Template Reference . . . . .	1395
4.599.1 Detailed Description . . . . .	1395
4.600std::hash< wstring > Struct Template Reference . . . . .	1395



4.600.1 Detailed Description . . . . .	1396
4.601std::hash<::vector< bool, _Alloc > > Struct Template Reference . . . . .	1396
4.601.1 Detailed Description . . . . .	1396
4.602std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > Class Template Reference . . . . .	1396
4.602.1 Detailed Description . . . . .	1397
4.602.2 Member Typedef Documentation . . . . .	1397
4.602.3 Constructor & Destructor Documentation . . . . .	1397
4.602.4 Member Function Documentation . . . . .	1398
4.602.5 Friends And Related Function Documentation . . . . .	1400
4.603std::indirect_array< _Tp > Class Template Reference . . . . .	1400
4.603.1 Detailed Description . . . . .	1401
4.604std::input_iterator_tag Struct Reference . . . . .	1402
4.604.1 Detailed Description . . . . .	1402
4.605std::insert_iterator< _Container > Class Template Reference . . . . .	1403
4.605.1 Detailed Description . . . . .	1403
4.605.2 Member Typedef Documentation . . . . .	1404
4.605.3 Constructor & Destructor Documentation . . . . .	1404
4.605.4 Member Function Documentation . . . . .	1404
4.606std::ios_base Class Reference . . . . .	1406
4.606.1 Detailed Description . . . . .	1408
4.606.2 Member Typedef Documentation . . . . .	1408
4.606.3 Member Enumeration Documentation . . . . .	1410
4.606.4 Constructor & Destructor Documentation . . . . .	1410
4.606.5 Member Function Documentation . . . . .	1411
4.606.6 Member Data Documentation . . . . .	1415
4.607std::ios_base::failure Class Reference . . . . .	1419
4.607.1 Detailed Description . . . . .	1419
4.608std::istream_iterator< _Tp, _CharT, _Traits, _Dist > Class Template Reference . . . . .	1419
4.608.1 Detailed Description . . . . .	1420
4.608.2 Member Typedef Documentation . . . . .	1420
4.608.3 Constructor & Destructor Documentation . . . . .	1421
4.609std::istreambuf_iterator< _CharT, _Traits > Class Template Reference . . . . .	1421
4.609.1 Detailed Description . . . . .	1422
4.609.2 Member Typedef Documentation . . . . .	1422
4.609.3 Constructor & Destructor Documentation . . . . .	1424
4.609.4 Member Function Documentation . . . . .	1424
4.610std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference > Struct Template Reference . . . . .	1425

4.610.1 Detailed Description . . . . .	1425
4.610.2 Member Typedef Documentation . . . . .	1425
4.611std::iterator_traits< _Tp * > Struct Template Reference . . . . .	1426
4.611.1 Detailed Description . . . . .	1426
4.612std::iterator_traits< const _Tp * > Struct Template Reference . . . . .	1426
4.612.1 Detailed Description . . . . .	1426
4.613std::less< _Tp > Struct Template Reference . . . . .	1427
4.613.1 Detailed Description . . . . .	1427
4.613.2 Member Typedef Documentation . . . . .	1427
4.614std::less_equal< _Tp > Struct Template Reference . . . . .	1428
4.614.1 Detailed Description . . . . .	1428
4.614.2 Member Typedef Documentation . . . . .	1428
4.615std::linear_congruential_engine< _UIntType, __a, __c, __m > Class Template Reference . . . . .	1429
4.615.1 Detailed Description . . . . .	1430
4.615.2 Member Typedef Documentation . . . . .	1430
4.615.3 Constructor & Destructor Documentation . . . . .	1430
4.615.4 Member Function Documentation . . . . .	1431
4.615.5 Friends And Related Function Documentation . . . . .	1432
4.615.6 Member Data Documentation . . . . .	1433
4.616std::list< _Tp, _Alloc > Class Template Reference . . . . .	1433
4.616.1 Detailed Description . . . . .	1436
4.616.2 Constructor & Destructor Documentation . . . . .	1437
4.616.3 Member Function Documentation . . . . .	1438
4.617std::locale Class Reference . . . . .	1451
4.617.1 Detailed Description . . . . .	1452
4.617.2 Member Typedef Documentation . . . . .	1453
4.617.3 Constructor & Destructor Documentation . . . . .	1453
4.617.4 Member Function Documentation . . . . .	1454
4.617.5 Member Data Documentation . . . . .	1456
4.618std::locale::facet Class Reference . . . . .	1458
4.618.1 Detailed Description . . . . .	1459
4.618.2 Constructor & Destructor Documentation . . . . .	1459
4.619std::locale::id Class Reference . . . . .	1459
4.619.1 Detailed Description . . . . .	1460
4.619.2 Constructor & Destructor Documentation . . . . .	1460
4.620std::logical_and< _Tp > Struct Template Reference . . . . .	1460
4.620.1 Detailed Description . . . . .	1461

4.620.2 Member Typedef Documentation . . . . .	1461
4.621std::logical_not< _Tp > Struct Template Reference . . . . .	1461
4.621.1 Detailed Description . . . . .	1462
4.621.2 Member Typedef Documentation . . . . .	1462
4.622std::logical_or< _Tp > Struct Template Reference . . . . .	1462
4.622.1 Detailed Description . . . . .	1463
4.622.2 Member Typedef Documentation . . . . .	1463
4.623std::lognormal_distribution< _RealType > Class Template Reference . . . . .	1463
4.623.1 Detailed Description . . . . .	1464
4.623.2 Member Typedef Documentation . . . . .	1464
4.623.3 Member Function Documentation . . . . .	1465
4.623.4 Friends And Related Function Documentation . . . . .	1465
4.624std::lognormal_distribution< _RealType >::param_type Struct Reference . . . . .	1466
4.624.1 Detailed Description . . . . .	1467
4.625std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference . . . . .	1467
4.625.1 Detailed Description . . . . .	1469
4.625.2 Constructor & Destructor Documentation . . . . .	1470
4.625.3 Member Function Documentation . . . . .	1471
4.626std::mask_array< _Tp > Class Template Reference . . . . .	1485
4.626.1 Detailed Description . . . . .	1486
4.627std::match_results< _Bi_iter, _Alloc > Class Template Reference . . . . .	1487
4.627.1 Detailed Description . . . . .	1490
4.627.2 Constructor & Destructor Documentation . . . . .	1491
4.627.3 Member Function Documentation . . . . .	1491
4.628std::mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference . . . . .	1496
4.628.1 Detailed Description . . . . .	1497
4.628.2 Member Typedef Documentation . . . . .	1497
4.629std::mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference . . . . .	1498
4.629.1 Detailed Description . . . . .	1498
4.629.2 Member Typedef Documentation . . . . .	1498
4.630std::mem_fun_ref_t< _Ret, _Tp > Class Template Reference . . . . .	1499
4.630.1 Detailed Description . . . . .	1499
4.630.2 Member Typedef Documentation . . . . .	1499
4.631std::mem_fun_t< _Ret, _Tp > Class Template Reference . . . . .	1500
4.631.1 Detailed Description . . . . .	1500
4.631.2 Member Typedef Documentation . . . . .	1501

4.632std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > Class Template Reference . . . . .	1501
4.632.1 Detailed Description . . . . .	1502
4.632.2 Member Typedef Documentation . . . . .	1503
4.632.3 Constructor & Destructor Documentation . . . . .	1503
4.632.4 Member Function Documentation . . . . .	1503
4.632.5 Friends And Related Function Documentation . . . . .	1504
4.633std::messages< _CharT > Class Template Reference . . . . .	1505
4.633.1 Detailed Description . . . . .	1506
4.633.2 Member Typedef Documentation . . . . .	1507
4.633.3 Constructor & Destructor Documentation . . . . .	1507
4.633.4 Member Data Documentation . . . . .	1507
4.634std::messages_base Struct Reference . . . . .	1508
4.634.1 Detailed Description . . . . .	1508
4.635std::messages_byname< _CharT > Class Template Reference . . . . .	1509
4.635.1 Detailed Description . . . . .	1510
4.635.2 Member Data Documentation . . . . .	1510
4.636std::minus< _Tp > Struct Template Reference . . . . .	1510
4.636.1 Detailed Description . . . . .	1511
4.636.2 Member Typedef Documentation . . . . .	1511
4.637std::modulus< _Tp > Struct Template Reference . . . . .	1512
4.637.1 Detailed Description . . . . .	1512
4.637.2 Member Typedef Documentation . . . . .	1512
4.638std::money_base Class Reference . . . . .	1513
4.638.1 Detailed Description . . . . .	1514
4.639std::money_get< _CharT, _InIter > Class Template Reference . . . . .	1514
4.639.1 Detailed Description . . . . .	1515
4.639.2 Member Typedef Documentation . . . . .	1515
4.639.3 Constructor & Destructor Documentation . . . . .	1516
4.639.4 Member Function Documentation . . . . .	1516
4.639.5 Member Data Documentation . . . . .	1518
4.640std::money_put< _CharT, _OutIter > Class Template Reference . . . . .	1518
4.640.1 Detailed Description . . . . .	1519
4.640.2 Member Typedef Documentation . . . . .	1519
4.640.3 Constructor & Destructor Documentation . . . . .	1520
4.640.4 Member Function Documentation . . . . .	1520
4.640.5 Member Data Documentation . . . . .	1523

4.641std::moneypunct< _CharT, _Intl > Class Template Reference . . . . .	1523
4.641.1 Detailed Description . . . . .	1525
4.641.2 Member Typedef Documentation . . . . .	1525
4.641.3 Constructor & Destructor Documentation . . . . .	1525
4.641.4 Member Function Documentation . . . . .	1526
4.641.5 Member Data Documentation . . . . .	1532
4.642std::moneypunct_byname< _CharT, _Intl > Class Template Reference . . . . .	1533
4.642.1 Detailed Description . . . . .	1534
4.642.2 Member Function Documentation . . . . .	1534
4.642.3 Member Data Documentation . . . . .	1540
4.643std::move_iterator< _Iterator > Class Template Reference . . . . .	1541
4.643.1 Detailed Description . . . . .	1541
4.644std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference . . . . .	1542
4.644.1 Detailed Description . . . . .	1543
4.644.2 Constructor & Destructor Documentation . . . . .	1544
4.644.3 Member Function Documentation . . . . .	1546
4.645std::multiplies< _Tp > Struct Template Reference . . . . .	1559
4.645.1 Detailed Description . . . . .	1560
4.645.2 Member Typedef Documentation . . . . .	1560
4.646std::multiset< _Key, _Compare, _Alloc > Class Template Reference . . . . .	1560
4.646.1 Detailed Description . . . . .	1562
4.646.2 Constructor & Destructor Documentation . . . . .	1562
4.646.3 Member Function Documentation . . . . .	1564
4.647std::negate< _Tp > Struct Template Reference . . . . .	1579
4.647.1 Detailed Description . . . . .	1579
4.647.2 Member Typedef Documentation . . . . .	1579
4.648std::negative_binomial_distribution< _IntType > Class Template Reference . . . . .	1580
4.648.1 Detailed Description . . . . .	1580
4.648.2 Member Typedef Documentation . . . . .	1581
4.648.3 Member Function Documentation . . . . .	1581
4.648.4 Friends And Related Function Documentation . . . . .	1582
4.649std::negative_binomial_distribution< _IntType >::param_type Struct Reference . . . . .	1583
4.649.1 Detailed Description . . . . .	1583
4.650std::nested_exception Class Reference . . . . .	1583
4.650.1 Detailed Description . . . . .	1583
4.651std::normal_distribution< _RealType > Class Template Reference . . . . .	1584
4.651.1 Detailed Description . . . . .	1584

4.651.2 Member Typedef Documentation . . . . .	1585
4.651.3 Constructor & Destructor Documentation . . . . .	1585
4.651.4 Member Function Documentation . . . . .	1585
4.651.5 Friends And Related Function Documentation . . . . .	1586
4.652std::normal_distribution< _RealType >::param_type Struct Reference . . . . .	1587
4.652.1 Detailed Description . . . . .	1587
4.653std::not_equal_to< _Tp > Struct Template Reference . . . . .	1588
4.653.1 Detailed Description . . . . .	1588
4.653.2 Member Typedef Documentation . . . . .	1588
4.654std::num_get< _CharT, _InIter > Class Template Reference . . . . .	1589
4.654.1 Detailed Description . . . . .	1590
4.654.2 Member Typedef Documentation . . . . .	1591
4.654.3 Constructor & Destructor Documentation . . . . .	1591
4.654.4 Member Function Documentation . . . . .	1591
4.654.5 Member Data Documentation . . . . .	1603
4.655std::num_put< _CharT, _OutIter > Class Template Reference . . . . .	1603
4.655.1 Detailed Description . . . . .	1605
4.655.2 Member Typedef Documentation . . . . .	1605
4.655.3 Constructor & Destructor Documentation . . . . .	1605
4.655.4 Member Function Documentation . . . . .	1605
4.655.5 Member Data Documentation . . . . .	1614
4.656std::num_punct< _CharT > Class Template Reference . . . . .	1615
4.656.1 Detailed Description . . . . .	1616
4.656.2 Member Typedef Documentation . . . . .	1616
4.656.3 Constructor & Destructor Documentation . . . . .	1617
4.656.4 Member Function Documentation . . . . .	1618
4.656.5 Member Data Documentation . . . . .	1621
4.657std::num_punct_byname< _CharT > Class Template Reference . . . . .	1622
4.657.1 Detailed Description . . . . .	1623
4.657.2 Member Function Documentation . . . . .	1623
4.657.3 Member Data Documentation . . . . .	1626
4.658std::ostream_iterator< _Tp, _CharT, _Traits > Class Template Reference . . . . .	1627
4.658.1 Detailed Description . . . . .	1627
4.658.2 Member Typedef Documentation . . . . .	1628
4.658.3 Constructor & Destructor Documentation . . . . .	1629
4.658.4 Member Function Documentation . . . . .	1629
4.659std::ostreambuf_iterator< _CharT, _Traits > Class Template Reference . . . . .	1630

4.659.1 Detailed Description . . . . .	1631
4.659.2 Member Typedef Documentation . . . . .	1631
4.659.3 Constructor & Destructor Documentation . . . . .	1632
4.659.4 Member Function Documentation . . . . .	1632
4.660std::output_iterator_tag Struct Reference . . . . .	1633
4.660.1 Detailed Description . . . . .	1633
4.661std::owner_less< _Tp > Struct Template Reference . . . . .	1633
4.661.1 Detailed Description . . . . .	1633
4.662std::owner_less< shared_ptr< _Tp > > Struct Template Reference . . . . .	1633
4.662.1 Detailed Description . . . . .	1634
4.662.2 Member Typedef Documentation . . . . .	1634
4.663std::owner_less< weak_ptr< _Tp > > Struct Template Reference . . . . .	1634
4.663.1 Detailed Description . . . . .	1634
4.663.2 Member Typedef Documentation . . . . .	1635
4.664std::pair< _T1, _T2 > Struct Template Reference . . . . .	1635
4.664.1 Detailed Description . . . . .	1636
4.664.2 Member Typedef Documentation . . . . .	1636
4.664.3 Constructor & Destructor Documentation . . . . .	1636
4.664.4 Member Data Documentation . . . . .	1637
4.665std::piecewise_constant_distribution< _RealType > Class Template Reference . . . . .	1637
4.665.1 Detailed Description . . . . .	1638
4.665.2 Member Typedef Documentation . . . . .	1638
4.665.3 Member Function Documentation . . . . .	1638
4.665.4 Friends And Related Function Documentation . . . . .	1640
4.666std::piecewise_constant_distribution< _RealType >::param_type Struct Reference . . . . .	1640
4.666.1 Detailed Description . . . . .	1641
4.667std::piecewise_construct_t Struct Reference . . . . .	1641
4.667.1 Detailed Description . . . . .	1641
4.668std::piecewise_linear_distribution< _RealType > Class Template Reference . . . . .	1641
4.668.1 Detailed Description . . . . .	1642
4.668.2 Member Typedef Documentation . . . . .	1642
4.668.3 Member Function Documentation . . . . .	1643
4.668.4 Friends And Related Function Documentation . . . . .	1644
4.669std::piecewise_linear_distribution< _RealType >::param_type Struct Reference . . . . .	1645
4.669.1 Detailed Description . . . . .	1645
4.670std::plus< _Tp > Struct Template Reference . . . . .	1645
4.670.1 Detailed Description . . . . .	1646

4.670.2 Member Typedef Documentation . . . . .	1646
4.671std::pointer_to_binary_function< _Arg1, _Arg2, _Result > Class Template Reference . . . . .	1647
4.671.1 Detailed Description . . . . .	1647
4.671.2 Member Typedef Documentation . . . . .	1647
4.672std::pointer_to_unary_function< _Arg, _Result > Class Template Reference . . . . .	1648
4.672.1 Detailed Description . . . . .	1649
4.672.2 Member Typedef Documentation . . . . .	1649
4.673std::pointer_traits< _Ptr > Struct Template Reference . . . . .	1649
4.673.1 Detailed Description . . . . .	1649
4.673.2 Member Typedef Documentation . . . . .	1650
4.674std::pointer_traits< _Tp * > Struct Template Reference . . . . .	1650
4.674.1 Detailed Description . . . . .	1650
4.674.2 Member Typedef Documentation . . . . .	1650
4.674.3 Member Function Documentation . . . . .	1651
4.675std::poisson_distribution< _IntType > Class Template Reference . . . . .	1651
4.675.1 Detailed Description . . . . .	1652
4.675.2 Member Typedef Documentation . . . . .	1652
4.675.3 Member Function Documentation . . . . .	1652
4.675.4 Friends And Related Function Documentation . . . . .	1653
4.676std::poisson_distribution< _IntType >::param_type Struct Reference . . . . .	1654
4.676.1 Detailed Description . . . . .	1654
4.677std::priority_queue< _Tp, _Sequence, _Compare > Class Template Reference . . . . .	1655
4.677.1 Detailed Description . . . . .	1655
4.677.2 Constructor & Destructor Documentation . . . . .	1656
4.677.3 Member Function Documentation . . . . .	1657
4.678std::queue< _Tp, _Sequence > Class Template Reference . . . . .	1658
4.678.1 Detailed Description . . . . .	1658
4.678.2 Constructor & Destructor Documentation . . . . .	1659
4.678.3 Member Function Documentation . . . . .	1659
4.678.4 Member Data Documentation . . . . .	1660
4.679std::random_access_iterator_tag Struct Reference . . . . .	1661
4.679.1 Detailed Description . . . . .	1661
4.680std::random_device Class Reference . . . . .	1661
4.680.1 Detailed Description . . . . .	1662
4.680.2 Member Typedef Documentation . . . . .	1662
4.681std::raw_storage_iterator< _OutputIterator, _Tp > Class Template Reference . . . . .	1662
4.681.1 Detailed Description . . . . .	1663



4.681.2 Member Typedef Documentation . . . . .	1663
4.682std::regex_error Class Reference . . . . .	1664
4.682.1 Detailed Description . . . . .	1664
4.682.2 Constructor & Destructor Documentation . . . . .	1664
4.682.3 Member Function Documentation . . . . .	1664
4.683std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference . . . . .	1664
4.683.1 Detailed Description . . . . .	1665
4.683.2 Constructor & Destructor Documentation . . . . .	1665
4.683.3 Member Function Documentation . . . . .	1666
4.684std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference . . . . .	1667
4.684.1 Detailed Description . . . . .	1668
4.684.2 Constructor & Destructor Documentation . . . . .	1668
4.684.3 Member Function Documentation . . . . .	1670
4.685std::regex_traits< _Ch_type > Struct Template Reference . . . . .	1672
4.685.1 Detailed Description . . . . .	1672
4.685.2 Constructor & Destructor Documentation . . . . .	1672
4.685.3 Member Function Documentation . . . . .	1673
4.686std::reverse_iterator< _Iterator > Class Template Reference . . . . .	1677
4.686.1 Detailed Description . . . . .	1678
4.686.2 Member Typedef Documentation . . . . .	1678
4.686.3 Constructor & Destructor Documentation . . . . .	1679
4.686.4 Member Function Documentation . . . . .	1679
4.687std::seed_seq Class Reference . . . . .	1682
4.687.1 Detailed Description . . . . .	1682
4.687.2 Member Typedef Documentation . . . . .	1682
4.687.3 Constructor & Destructor Documentation . . . . .	1682
4.688std::set< _Key, _Compare, _Alloc > Class Template Reference . . . . .	1683
4.688.1 Detailed Description . . . . .	1684
4.688.2 Member Typedef Documentation . . . . .	1685
4.688.3 Constructor & Destructor Documentation . . . . .	1686
4.688.4 Member Function Documentation . . . . .	1689
4.689std::shared_ptr< _Tp > Class Template Reference . . . . .	1702
4.689.1 Detailed Description . . . . .	1703
4.689.2 Constructor & Destructor Documentation . . . . .	1704
4.689.3 Friends And Related Function Documentation . . . . .	1707
4.690std::shuffle_order_engine< _RandomNumberEngine, __k > Class Template Reference . . . . .	1709
4.690.1 Detailed Description . . . . .	1710

4.690.2 Member Typedef Documentation . . . . .	1710
4.690.3 Constructor & Destructor Documentation . . . . .	1710
4.690.4 Member Function Documentation . . . . .	1712
4.690.5 Friends And Related Function Documentation . . . . .	1713
4.691std::slice Class Reference . . . . .	1714
4.691.1 Detailed Description . . . . .	1714
4.692std::slice_array< _Tp > Class Template Reference . . . . .	1715
4.692.1 Detailed Description . . . . .	1715
4.693std::stack< _Tp, _Sequence > Class Template Reference . . . . .	1716
4.693.1 Detailed Description . . . . .	1716
4.693.2 Constructor & Destructor Documentation . . . . .	1717
4.693.3 Member Function Documentation . . . . .	1717
4.694std::student_t_distribution< _RealType > Class Template Reference . . . . .	1718
4.694.1 Detailed Description . . . . .	1719
4.694.2 Member Typedef Documentation . . . . .	1719
4.694.3 Member Function Documentation . . . . .	1719
4.694.4 Friends And Related Function Documentation . . . . .	1720
4.695std::student_t_distribution< _RealType >::param_type Struct Reference . . . . .	1721
4.695.1 Detailed Description . . . . .	1721
4.696std::sub_match< _Biter > Class Template Reference . . . . .	1722
4.696.1 Detailed Description . . . . .	1722
4.696.2 Member Typedef Documentation . . . . .	1723
4.696.3 Member Function Documentation . . . . .	1723
4.696.4 Member Data Documentation . . . . .	1724
4.697std::time_base Class Reference . . . . .	1725
4.697.1 Detailed Description . . . . .	1725
4.698std::time_get< _CharT, _InIter > Class Template Reference . . . . .	1726
4.698.1 Detailed Description . . . . .	1727
4.698.2 Member Typedef Documentation . . . . .	1727
4.698.3 Constructor & Destructor Documentation . . . . .	1728
4.698.4 Member Function Documentation . . . . .	1728
4.698.5 Member Data Documentation . . . . .	1734
4.699std::time_get_byname< _CharT, _InIter > Class Template Reference . . . . .	1734
4.699.1 Detailed Description . . . . .	1736
4.699.2 Member Function Documentation . . . . .	1736
4.699.3 Member Data Documentation . . . . .	1741
4.700std::time_put< _CharT, _OutIter > Class Template Reference . . . . .	1742

4.700.1 Detailed Description . . . . .	1743
4.700.2 Member Typedef Documentation . . . . .	1743
4.700.3 Constructor & Destructor Documentation . . . . .	1743
4.700.4 Member Function Documentation . . . . .	1744
4.700.5 Member Data Documentation . . . . .	1745
4.701std::time_put_byname< _CharT, _OutIter > Class Template Reference . . . . .	1745
4.701.1 Detailed Description . . . . .	1746
4.701.2 Member Function Documentation . . . . .	1746
4.701.3 Member Data Documentation . . . . .	1748
4.702std::unary_function< _Arg, _Result > Struct Template Reference . . . . .	1748
4.702.1 Detailed Description . . . . .	1749
4.702.2 Member Typedef Documentation . . . . .	1749
4.703std::unary_negate< _Predicate > Class Template Reference . . . . .	1749
4.703.1 Detailed Description . . . . .	1750
4.703.2 Member Typedef Documentation . . . . .	1750
4.704std::uniform_int_distribution< _IntType > Class Template Reference . . . . .	1750
4.704.1 Detailed Description . . . . .	1751
4.704.2 Member Typedef Documentation . . . . .	1751
4.704.3 Constructor & Destructor Documentation . . . . .	1751
4.704.4 Member Function Documentation . . . . .	1751
4.704.5 Friends And Related Function Documentation . . . . .	1752
4.705std::uniform_int_distribution< _IntType >::param_type Struct Reference . . . . .	1753
4.705.1 Detailed Description . . . . .	1753
4.706std::uniform_real_distribution< _RealType > Class Template Reference . . . . .	1753
4.706.1 Detailed Description . . . . .	1754
4.706.2 Member Typedef Documentation . . . . .	1754
4.706.3 Constructor & Destructor Documentation . . . . .	1754
4.706.4 Member Function Documentation . . . . .	1754
4.706.5 Friends And Related Function Documentation . . . . .	1755
4.707std::uniform_real_distribution< _RealType >::param_type Struct Reference . . . . .	1756
4.707.1 Detailed Description . . . . .	1756
4.708std::unique_ptr< _Tp, _Dp > Class Template Reference . . . . .	1756
4.708.1 Detailed Description . . . . .	1757
4.709std::unique_ptr< _Tp[], _Dp > Class Template Reference . . . . .	1757
4.709.1 Detailed Description . . . . .	1758
4.710std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference . . . . .	1758
4.710.1 Detailed Description . . . . .	1760

4.710.2 Member Typedef Documentation . . . . .	1761
4.710.3 Constructor & Destructor Documentation . . . . .	1763
4.710.4 Member Function Documentation . . . . .	1764
4.711std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference . . . . .	1778
4.711.1 Detailed Description . . . . .	1780
4.711.2 Member Typedef Documentation . . . . .	1781
4.711.3 Constructor & Destructor Documentation . . . . .	1783
4.711.4 Member Function Documentation . . . . .	1784
4.712std::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference . . . . .	1800
4.712.1 Detailed Description . . . . .	1802
4.712.2 Member Typedef Documentation . . . . .	1802
4.712.3 Constructor & Destructor Documentation . . . . .	1804
4.712.4 Member Function Documentation . . . . .	1807
4.713std::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference . . . . .	1820
4.713.1 Detailed Description . . . . .	1822
4.713.2 Member Typedef Documentation . . . . .	1822
4.713.3 Constructor & Destructor Documentation . . . . .	1824
4.713.4 Member Function Documentation . . . . .	1825
4.714std::uses_allocator< _Tp, _Alloc > Struct Template Reference . . . . .	1840
4.714.1 Detailed Description . . . . .	1840
4.715std::vector< _Tp, _Alloc > Class Template Reference . . . . .	1841
4.715.1 Detailed Description . . . . .	1844
4.715.2 Constructor & Destructor Documentation . . . . .	1844
4.715.3 Member Function Documentation . . . . .	1846
4.716std::vector< bool, _Alloc > Class Template Reference . . . . .	1859
4.716.1 Detailed Description . . . . .	1861
4.717std::weak_ptr< _Tp > Class Template Reference . . . . .	1862
4.717.1 Detailed Description . . . . .	1862
4.718std::weibull_distribution< _RealType > Class Template Reference . . . . .	1863
4.718.1 Detailed Description . . . . .	1863
4.718.2 Member Typedef Documentation . . . . .	1864
4.718.3 Member Function Documentation . . . . .	1864
4.718.4 Friends And Related Function Documentation . . . . .	1865
4.719std::weibull_distribution< _RealType >::param_type Struct Reference . . . . .	1865
4.719.1 Detailed Description . . . . .	1865

5.1	<a href="#">algo.h File Reference</a>	1866
5.1.1	<a href="#">Detailed Description</a>	1874
5.2	<a href="#">algbase.h File Reference</a>	1874
5.2.1	<a href="#">Detailed Description</a>	1875
5.3	<a href="#">algorithmfwd.h File Reference</a>	1875
5.3.1	<a href="#">Detailed Description</a>	1878
5.4	<a href="#">algorithmfwd.h File Reference</a>	1879
5.4.1	<a href="#">Detailed Description</a>	1886
5.5	<a href="#">alloc_traits.h File Reference</a>	1886
5.5.1	<a href="#">Detailed Description</a>	1886
5.6	<a href="#">alloc_traits.h File Reference</a>	1887
5.6.1	<a href="#">Detailed Description</a>	1887
5.7	<a href="#">allocator.h File Reference</a>	1887
5.7.1	<a href="#">Detailed Description</a>	1887
5.8	<a href="#">array_allocator.h File Reference</a>	1887
5.8.1	<a href="#">Detailed Description</a>	1888
5.9	<a href="#">assoc_container.hpp File Reference</a>	1888
5.9.1	<a href="#">Detailed Description</a>	1888
5.10	<a href="#">atomic_base.h File Reference</a>	1889
5.10.1	<a href="#">Detailed Description</a>	1890
5.11	<a href="#">atomic_lockfree_defines.h File Reference</a>	1890
5.11.1	<a href="#">Detailed Description</a>	1891
5.12	<a href="#">atomic_word.h File Reference</a>	1891
5.12.1	<a href="#">Detailed Description</a>	1891
5.13	<a href="#">atomicity.h File Reference</a>	1891
5.13.1	<a href="#">Detailed Description</a>	1891
5.14	<a href="#">auto_ptr.h File Reference</a>	1892
5.14.1	<a href="#">Detailed Description</a>	1892
5.15	<a href="#">backward_warning.h File Reference</a>	1892
5.15.1	<a href="#">Detailed Description</a>	1892
5.16	<a href="#">balanced_quicksort.h File Reference</a>	1892
5.16.1	<a href="#">Detailed Description</a>	1893
5.17	<a href="#">base.h File Reference</a>	1893
5.17.1	<a href="#">Detailed Description</a>	1893
5.18	<a href="#">base.h File Reference</a>	1893
5.18.1	<a href="#">Detailed Description</a>	1894
5.19	<a href="#">basic_file.h File Reference</a>	1894

5.19.1 Detailed Description . . . . .	1894
5.20 basic_ios.h File Reference . . . . .	1894
5.20.1 Detailed Description . . . . .	1894
5.21 basic_iterator.h File Reference . . . . .	1895
5.21.1 Detailed Description . . . . .	1895
5.22 basic_string.h File Reference . . . . .	1895
5.22.1 Detailed Description . . . . .	1897
5.23 bin_search_tree_.hpp File Reference . . . . .	1897
5.23.1 Detailed Description . . . . .	1897
5.24 binary_heap_.hpp File Reference . . . . .	1897
5.24.1 Detailed Description . . . . .	1897
5.25 binders.h File Reference . . . . .	1898
5.25.1 Detailed Description . . . . .	1898
5.26 binomial_heap_.hpp File Reference . . . . .	1898
5.26.1 Detailed Description . . . . .	1898
5.27 binomial_heap_base_.hpp File Reference . . . . .	1898
5.27.1 Detailed Description . . . . .	1899
5.28 bitmap_allocator.h File Reference . . . . .	1899
5.28.1 Detailed Description . . . . .	1900
5.28.2 Macro Definition Documentation . . . . .	1900
5.29 boost_concept_check.h File Reference . . . . .	1900
5.29.1 Detailed Description . . . . .	1900
5.30 branch_policy.hpp File Reference . . . . .	1901
5.30.1 Detailed Description . . . . .	1901
5.31 c++0x_warning.h File Reference . . . . .	1901
5.31.1 Detailed Description . . . . .	1901
5.32 c++allocator.h File Reference . . . . .	1901
5.32.1 Detailed Description . . . . .	1901
5.33 c++config.h File Reference . . . . .	1901
5.33.1 Detailed Description . . . . .	1906
5.34 c++io.h File Reference . . . . .	1906
5.34.1 Detailed Description . . . . .	1906
5.35 c++locale.h File Reference . . . . .	1906
5.35.1 Detailed Description . . . . .	1907
5.36 c++locale_internal.h File Reference . . . . .	1907
5.36.1 Detailed Description . . . . .	1907
5.37 cast.h File Reference . . . . .	1907

5.37.1 Detailed Description . . . . .	1908
5.38 cc_hash_max_collision_check_resize_trigger_imp.hpp File Reference . . . . .	1908
5.38.1 Detailed Description . . . . .	1908
5.39 cc_ht_map_.hpp File Reference . . . . .	1908
5.39.1 Detailed Description . . . . .	1908
5.40 char_traits.h File Reference . . . . .	1908
5.40.1 Detailed Description . . . . .	1909
5.41 checkers.h File Reference . . . . .	1909
5.41.1 Detailed Description . . . . .	1909
5.42 cmp_fn_imps.hpp File Reference . . . . .	1909
5.42.1 Detailed Description . . . . .	1909
5.43 codecvt.h File Reference . . . . .	1909
5.43.1 Detailed Description . . . . .	1910
5.44 codecvt_specializations.h File Reference . . . . .	1910
5.44.1 Detailed Description . . . . .	1910
5.45 compatibility.h File Reference . . . . .	1910
5.45.1 Detailed Description . . . . .	1910
5.46 compatibility.h File Reference . . . . .	1910
5.46.1 Detailed Description . . . . .	1911
5.47 compiletime_settings.h File Reference . . . . .	1911
5.47.1 Detailed Description . . . . .	1911
5.47.2 Macro Definition Documentation . . . . .	1911
5.48 complex.h File Reference . . . . .	1912
5.48.1 Detailed Description . . . . .	1912
5.49 concept_check.h File Reference . . . . .	1912
5.49.1 Detailed Description . . . . .	1913
5.50 concurrence.h File Reference . . . . .	1913
5.50.1 Detailed Description . . . . .	1913
5.51 cond_dealtor.hpp File Reference . . . . .	1913
5.51.1 Detailed Description . . . . .	1913
5.52 cond_key_dtor_entry_dealtor.hpp File Reference . . . . .	1914
5.52.1 Detailed Description . . . . .	1914
5.53 const_iterator.hpp File Reference . . . . .	1914
5.53.1 Detailed Description . . . . .	1914
5.54 const_iterator.hpp File Reference . . . . .	1914
5.54.1 Detailed Description . . . . .	1915
5.55 const_iterator.hpp File Reference . . . . .	1915

5.55.1 Detailed Description . . . . .	1915
5.56 constructor_destructor_fn_imps.hpp File Reference . . . . .	1915
5.56.1 Detailed Description . . . . .	1915
5.57 constructor_destructor_fn_imps.hpp File Reference . . . . .	1915
5.57.1 Detailed Description . . . . .	1915
5.58 constructor_destructor_fn_imps.hpp File Reference . . . . .	1915
5.59 constructor_destructor_no_store_hash_fn_imps.hpp File Reference . . . . .	1915
5.59.1 Detailed Description . . . . .	1915
5.60 constructor_destructor_no_store_hash_fn_imps.hpp File Reference . . . . .	1915
5.60.1 Detailed Description . . . . .	1915
5.61 constructor_destructor_store_hash_fn_imps.hpp File Reference . . . . .	1915
5.61.1 Detailed Description . . . . .	1915
5.62 constructor_destructor_store_hash_fn_imps.hpp File Reference . . . . .	1916
5.62.1 Detailed Description . . . . .	1916
5.63 constructors_destructor_fn_imps.hpp File Reference . . . . .	1916
5.63.1 Detailed Description . . . . .	1916
5.64 constructors_destructor_fn_imps.hpp File Reference . . . . .	1916
5.64.1 Detailed Description . . . . .	1916
5.65 constructors_destructor_fn_imps.hpp File Reference . . . . .	1916
5.65.1 Detailed Description . . . . .	1916
5.66 constructors_destructor_fn_imps.hpp File Reference . . . . .	1916
5.66.1 Detailed Description . . . . .	1916
5.67 constructors_destructor_fn_imps.hpp File Reference . . . . .	1916
5.67.1 Detailed Description . . . . .	1916
5.68 constructors_destructor_fn_imps.hpp File Reference . . . . .	1916
5.68.1 Detailed Description . . . . .	1916
5.69 constructors_destructor_fn_imps.hpp File Reference . . . . .	1916
5.69.1 Detailed Description . . . . .	1916
5.70 constructors_destructor_fn_imps.hpp File Reference . . . . .	1917
5.70.1 Detailed Description . . . . .	1917
5.71 constructors_destructor_fn_imps.hpp File Reference . . . . .	1917
5.71.1 Detailed Description . . . . .	1917
5.72 constructors_destructor_fn_imps.hpp File Reference . . . . .	1917
5.72.1 Detailed Description . . . . .	1917
5.73 constructors_destructor_fn_imps.hpp File Reference . . . . .	1917
5.73.1 Detailed Description . . . . .	1917
5.74 constructors_destructor_fn_imps.hpp File Reference . . . . .	1917



5.74.1 Detailed Description . . . . .	1917
5.75 container_base_dispatch.hpp File Reference . . . . .	1917
5.75.1 Detailed Description . . . . .	1918
5.76 cpp_type_traits.h File Reference . . . . .	1918
5.76.1 Detailed Description . . . . .	1918
5.77 cpu_defines.h File Reference . . . . .	1918
5.77.1 Detailed Description . . . . .	1918
5.78 ctype_base.h File Reference . . . . .	1919
5.78.1 Detailed Description . . . . .	1919
5.79 ctype_inline.h File Reference . . . . .	1919
5.79.1 Detailed Description . . . . .	1919
5.80 cxxabi.h File Reference . . . . .	1919
5.80.1 Detailed Description . . . . .	1920
5.81 cxxabi_forced.h File Reference . . . . .	1921
5.81.1 Detailed Description . . . . .	1921
5.82 cxxabi_tweaks.h File Reference . . . . .	1921
5.82.1 Detailed Description . . . . .	1921
5.83 debug.h File Reference . . . . .	1921
5.83.1 Detailed Description . . . . .	1922
5.84 debug_allocator.h File Reference . . . . .	1922
5.84.1 Detailed Description . . . . .	1922
5.85 debug_fn_imps.hpp File Reference . . . . .	1922
5.85.1 Detailed Description . . . . .	1922
5.86 debug_fn_imps.hpp File Reference . . . . .	1923
5.86.1 Detailed Description . . . . .	1923
5.87 debug_fn_imps.hpp File Reference . . . . .	1923
5.87.1 Detailed Description . . . . .	1923
5.88 debug_fn_imps.hpp File Reference . . . . .	1923
5.88.1 Detailed Description . . . . .	1923
5.89 debug_fn_imps.hpp File Reference . . . . .	1923
5.89.1 Detailed Description . . . . .	1923
5.90 debug_fn_imps.hpp File Reference . . . . .	1923
5.90.1 Detailed Description . . . . .	1923
5.91 debug_fn_imps.hpp File Reference . . . . .	1923
5.91.1 Detailed Description . . . . .	1923
5.92 debug_fn_imps.hpp File Reference . . . . .	1923
5.92.1 Detailed Description . . . . .	1923

5.93 debug_fn_imps.hpp File Reference . . . . .	1923
5.93.1 Detailed Description . . . . .	1923
5.94 debug_fn_imps.hpp File Reference . . . . .	1924
5.94.1 Detailed Description . . . . .	1924
5.95 debug_fn_imps.hpp File Reference . . . . .	1924
5.95.1 Detailed Description . . . . .	1924
5.96 debug_fn_imps.hpp File Reference . . . . .	1924
5.96.1 Detailed Description . . . . .	1924
5.97 debug_fn_imps.hpp File Reference . . . . .	1924
5.97.1 Detailed Description . . . . .	1924
5.98 debug_fn_imps.hpp File Reference . . . . .	1924
5.98.1 Detailed Description . . . . .	1924
5.99 debug_fn_imps.hpp File Reference . . . . .	1924
5.99.1 Detailed Description . . . . .	1924
5.100 debug_map_base.hpp File Reference . . . . .	1924
5.100.1 Detailed Description . . . . .	1924
5.101 debug_no_store_hash_fn_imps.hpp File Reference . . . . .	1924
5.101.1 Detailed Description . . . . .	1924
5.102 debug_no_store_hash_fn_imps.hpp File Reference . . . . .	1925
5.102.1 Detailed Description . . . . .	1925
5.103 debug_store_hash_fn_imps.hpp File Reference . . . . .	1925
5.103.1 Detailed Description . . . . .	1925
5.104 debug_store_hash_fn_imps.hpp File Reference . . . . .	1925
5.104.1 Detailed Description . . . . .	1925
5.105 direct_mask_range_hashing_imp.hpp File Reference . . . . .	1925
5.105.1 Detailed Description . . . . .	1925
5.106 direct_mod_range_hashing_imp.hpp File Reference . . . . .	1925
5.106.1 Detailed Description . . . . .	1925
5.107 enc_filebuf.h File Reference . . . . .	1925
5.107.1 Detailed Description . . . . .	1925
5.108 entry_cmp.hpp File Reference . . . . .	1926
5.108.1 Detailed Description . . . . .	1926
5.109 entry_list_fn_imps.hpp File Reference . . . . .	1926
5.109.1 Detailed Description . . . . .	1926
5.110 entry_metadata_base.hpp File Reference . . . . .	1926
5.110.1 Detailed Description . . . . .	1926
5.111 entry_pred.hpp File Reference . . . . .	1926

5.111.1 Detailed Description . . . . .	1926
5.112eq_by_less.hpp File Reference . . . . .	1927
5.112.1 Detailed Description . . . . .	1927
5.113equally_split.h File Reference . . . . .	1927
5.113.1 Detailed Description . . . . .	1927
5.114erase_fn_imps.hpp File Reference . . . . .	1927
5.114.1 Detailed Description . . . . .	1927
5.115erase_fn_imps.hpp File Reference . . . . .	1927
5.115.1 Detailed Description . . . . .	1927
5.116erase_fn_imps.hpp File Reference . . . . .	1927
5.116.1 Detailed Description . . . . .	1927
5.117erase_fn_imps.hpp File Reference . . . . .	1928
5.117.1 Detailed Description . . . . .	1928
5.118erase_fn_imps.hpp File Reference . . . . .	1928
5.118.1 Detailed Description . . . . .	1928
5.119erase_fn_imps.hpp File Reference . . . . .	1928
5.119.1 Detailed Description . . . . .	1928
5.120erase_fn_imps.hpp File Reference . . . . .	1928
5.120.1 Detailed Description . . . . .	1928
5.121erase_fn_imps.hpp File Reference . . . . .	1928
5.121.1 Detailed Description . . . . .	1928
5.122erase_fn_imps.hpp File Reference . . . . .	1928
5.122.1 Detailed Description . . . . .	1928
5.123erase_fn_imps.hpp File Reference . . . . .	1928
5.123.1 Detailed Description . . . . .	1928
5.124erase_fn_imps.hpp File Reference . . . . .	1928
5.124.1 Detailed Description . . . . .	1928
5.125erase_fn_imps.hpp File Reference . . . . .	1929
5.125.1 Detailed Description . . . . .	1929
5.126erase_fn_imps.hpp File Reference . . . . .	1929
5.126.1 Detailed Description . . . . .	1929
5.127erase_fn_imps.hpp File Reference . . . . .	1929
5.127.1 Detailed Description . . . . .	1929
5.128erase_no_store_hash_fn_imps.hpp File Reference . . . . .	1929
5.128.1 Detailed Description . . . . .	1929
5.129erase_no_store_hash_fn_imps.hpp File Reference . . . . .	1929
5.129.1 Detailed Description . . . . .	1929

5.130erase_store_hash_fn_imps.hpp File Reference . . . . .	1929
5.130.1 Detailed Description . . . . .	1929
5.131erase_store_hash_fn_imps.hpp File Reference . . . . .	1929
5.131.1 Detailed Description . . . . .	1929
5.132error_constants.h File Reference . . . . .	1929
5.132.1 Detailed Description . . . . .	1930
5.133exception.hpp File Reference . . . . .	1930
5.133.1 Detailed Description . . . . .	1930
5.134exception_defines.h File Reference . . . . .	1931
5.134.1 Detailed Description . . . . .	1931
5.135exception_ptr.h File Reference . . . . .	1931
5.135.1 Detailed Description . . . . .	1931
5.136extc++.h File Reference . . . . .	1931
5.136.1 Detailed Description . . . . .	1931
5.137extptr_allocator.h File Reference . . . . .	1932
5.137.1 Detailed Description . . . . .	1932
5.138features.h File Reference . . . . .	1932
5.138.1 Detailed Description . . . . .	1932
5.138.2 Macro Definition Documentation . . . . .	1933
5.139fenv.h File Reference . . . . .	1934
5.139.1 Detailed Description . . . . .	1934
5.140find.h File Reference . . . . .	1934
5.140.1 Detailed Description . . . . .	1935
5.141find_fn_imps.hpp File Reference . . . . .	1935
5.141.1 Detailed Description . . . . .	1935
5.142find_fn_imps.hpp File Reference . . . . .	1935
5.142.1 Detailed Description . . . . .	1935
5.143find_fn_imps.hpp File Reference . . . . .	1935
5.143.1 Detailed Description . . . . .	1935
5.144find_fn_imps.hpp File Reference . . . . .	1935
5.144.1 Detailed Description . . . . .	1935
5.145find_fn_imps.hpp File Reference . . . . .	1935
5.145.1 Detailed Description . . . . .	1935
5.146find_fn_imps.hpp File Reference . . . . .	1935
5.146.1 Detailed Description . . . . .	1935
5.147find_fn_imps.hpp File Reference . . . . .	1936
5.147.1 Detailed Description . . . . .	1936

5.148	<a href="#">find_fn_imps.hpp File Reference</a>	1936
5.148.1	Detailed Description	1936
5.149	<a href="#">find_fn_imps.hpp File Reference</a>	1936
5.149.1	Detailed Description	1936
5.150	<a href="#">find_fn_imps.hpp File Reference</a>	1936
5.150.1	Detailed Description	1936
5.151	<a href="#">find_fn_imps.hpp File Reference</a>	1936
5.151.1	Detailed Description	1936
5.152	<a href="#">find_no_store_hash_fn_imps.hpp File Reference</a>	1936
5.152.1	Detailed Description	1936
5.153	<a href="#">find_selectors.h File Reference</a>	1936
5.153.1	Detailed Description	1937
5.154	<a href="#">find_store_hash_fn_imps.hpp File Reference</a>	1937
5.154.1	Detailed Description	1937
5.155	<a href="#">find_store_hash_fn_imps.hpp File Reference</a>	1937
5.155.1	Detailed Description	1937
5.156	<a href="#">for_each.h File Reference</a>	1937
5.156.1	Detailed Description	1937
5.157	<a href="#">for_each_selectors.h File Reference</a>	1937
5.157.1	Detailed Description	1938
5.158	<a href="#">formatter.h File Reference</a>	1938
5.158.1	Detailed Description	1939
5.159	<a href="#">forward_list.h File Reference</a>	1939
5.159.1	Detailed Description	1940
5.160	<a href="#">functexcept.h File Reference</a>	1940
5.160.1	Detailed Description	1940
5.161	<a href="#">functional_hash.h File Reference</a>	1940
5.161.1	Detailed Description	1941
5.162	<a href="#">functions.h File Reference</a>	1941
5.162.1	Detailed Description	1943
5.163	<a href="#">gp_ht_map_.hpp File Reference</a>	1943
5.163.1	Detailed Description	1943
5.164	<a href="#">gslice.h File Reference</a>	1943
5.164.1	Detailed Description	1944
5.165	<a href="#">gslice_array.h File Reference</a>	1944
5.165.1	Detailed Description	1944
5.166	<a href="#">hash_bytes.h File Reference</a>	1944

5.166.1 Detailed Description . . . . .	1944
5.167hash_eq_fn.hpp File Reference . . . . .	1945
5.167.1 Detailed Description . . . . .	1945
5.168hash_exponential_size_policy_imp.hpp File Reference . . . . .	1945
5.168.1 Detailed Description . . . . .	1945
5.169hash_fun.h File Reference . . . . .	1945
5.169.1 Detailed Description . . . . .	1945
5.170hash_load_check_resize_trigger_imp.hpp File Reference . . . . .	1945
5.170.1 Detailed Description . . . . .	1945
5.171hash_load_check_resize_trigger_size_base.hpp File Reference . . . . .	1946
5.171.1 Detailed Description . . . . .	1946
5.172hash_policy.hpp File Reference . . . . .	1946
5.172.1 Detailed Description . . . . .	1947
5.173hash_prime_size_policy_imp.hpp File Reference . . . . .	1947
5.173.1 Detailed Description . . . . .	1947
5.174hash_standard_resize_policy_imp.hpp File Reference . . . . .	1947
5.174.1 Detailed Description . . . . .	1947
5.175hashtable.h File Reference . . . . .	1947
5.175.1 Detailed Description . . . . .	1948
5.176hashtable.h File Reference . . . . .	1948
5.176.1 Detailed Description . . . . .	1948
5.177hashtable_policy.h File Reference . . . . .	1948
5.177.1 Detailed Description . . . . .	1950
5.178indirect_array.h File Reference . . . . .	1950
5.178.1 Detailed Description . . . . .	1950
5.179info_fn_imps.hpp File Reference . . . . .	1951
5.179.1 Detailed Description . . . . .	1951
5.180info_fn_imps.hpp File Reference . . . . .	1951
5.180.1 Detailed Description . . . . .	1951
5.181info_fn_imps.hpp File Reference . . . . .	1951
5.181.1 Detailed Description . . . . .	1951
5.182info_fn_imps.hpp File Reference . . . . .	1951
5.182.1 Detailed Description . . . . .	1951
5.183info_fn_imps.hpp File Reference . . . . .	1951
5.183.1 Detailed Description . . . . .	1951
5.184info_fn_imps.hpp File Reference . . . . .	1951
5.184.1 Detailed Description . . . . .	1951

5.185info_fn_imps.hpp File Reference . . . . .	1951
5.185.1 Detailed Description . . . . .	1951
5.186info_fn_imps.hpp File Reference . . . . .	1951
5.186.1 Detailed Description . . . . .	1951
5.187info_fn_imps.hpp File Reference . . . . .	1952
5.187.1 Detailed Description . . . . .	1952
5.188info_fn_imps.hpp File Reference . . . . .	1952
5.188.1 Detailed Description . . . . .	1952
5.189insert_fn_imps.hpp File Reference . . . . .	1952
5.189.1 Detailed Description . . . . .	1952
5.190insert_fn_imps.hpp File Reference . . . . .	1952
5.190.1 Detailed Description . . . . .	1952
5.191insert_fn_imps.hpp File Reference . . . . .	1952
5.191.1 Detailed Description . . . . .	1952
5.192insert_fn_imps.hpp File Reference . . . . .	1952
5.192.1 Detailed Description . . . . .	1952
5.193insert_fn_imps.hpp File Reference . . . . .	1952
5.193.1 Detailed Description . . . . .	1952
5.194insert_fn_imps.hpp File Reference . . . . .	1952
5.194.1 Detailed Description . . . . .	1952
5.195insert_fn_imps.hpp File Reference . . . . .	1953
5.195.1 Detailed Description . . . . .	1953
5.196insert_fn_imps.hpp File Reference . . . . .	1953
5.196.1 Detailed Description . . . . .	1953
5.197insert_fn_imps.hpp File Reference . . . . .	1953
5.197.1 Detailed Description . . . . .	1953
5.198insert_fn_imps.hpp File Reference . . . . .	1953
5.198.1 Detailed Description . . . . .	1953
5.199insert_fn_imps.hpp File Reference . . . . .	1953
5.199.1 Detailed Description . . . . .	1953
5.200insert_fn_imps.hpp File Reference . . . . .	1953
5.200.1 Detailed Description . . . . .	1953
5.201insert_fn_imps.hpp File Reference . . . . .	1953
5.201.1 Detailed Description . . . . .	1953
5.202insert_join_fn_imps.hpp File Reference . . . . .	1953
5.202.1 Detailed Description . . . . .	1953
5.203insert_no_store_hash_fn_imps.hpp File Reference . . . . .	1954

5.203.1 Detailed Description . . . . .	1954
5.204insert_no_store_hash_fn_imps.hpp File Reference . . . . .	1954
5.204.1 Detailed Description . . . . .	1954
5.205insert_store_hash_fn_imps.hpp File Reference . . . . .	1954
5.205.1 Detailed Description . . . . .	1954
5.206insert_store_hash_fn_imps.hpp File Reference . . . . .	1954
5.206.1 Detailed Description . . . . .	1954
5.207ios_base.h File Reference . . . . .	1954
5.207.1 Detailed Description . . . . .	1955
5.208iterator.h File Reference . . . . .	1956
5.208.1 Detailed Description . . . . .	1956
5.209iterator.hpp File Reference . . . . .	1956
5.209.1 Detailed Description . . . . .	1956
5.210iterator_fn_imps.hpp File Reference . . . . .	1956
5.210.1 Detailed Description . . . . .	1956
5.211iterator_tracker.h File Reference . . . . .	1956
5.211.1 Detailed Description . . . . .	1957
5.212iterators_fn_imps.hpp File Reference . . . . .	1957
5.212.1 Detailed Description . . . . .	1957
5.213iterators_fn_imps.hpp File Reference . . . . .	1957
5.213.1 Detailed Description . . . . .	1957
5.214iterators_fn_imps.hpp File Reference . . . . .	1957
5.214.1 Detailed Description . . . . .	1957
5.215iterators_fn_imps.hpp File Reference . . . . .	1958
5.215.1 Detailed Description . . . . .	1958
5.216iterators_fn_imps.hpp File Reference . . . . .	1958
5.216.1 Detailed Description . . . . .	1958
5.217iterators_fn_imps.hpp File Reference . . . . .	1958
5.217.1 Detailed Description . . . . .	1958
5.218iterators_fn_imps.hpp File Reference . . . . .	1958
5.218.1 Detailed Description . . . . .	1958
5.219left_child_next_sibling_heap_.hpp File Reference . . . . .	1958
5.219.1 Detailed Description . . . . .	1958
5.220linear_probe_fn_imp.hpp File Reference . . . . .	1958
5.220.1 Detailed Description . . . . .	1958
5.221list_partition.h File Reference . . . . .	1959
5.221.1 Detailed Description . . . . .	1959



5.222list_update_policy.hpp File Reference . . . . .	1959
5.222.1 Detailed Description . . . . .	1959
5.223locale_classes.h File Reference . . . . .	1959
5.223.1 Detailed Description . . . . .	1960
5.224locale_facets.h File Reference . . . . .	1960
5.224.1 Detailed Description . . . . .	1961
5.225locale_facets_nonio.h File Reference . . . . .	1961
5.225.1 Detailed Description . . . . .	1961
5.226localefwd.h File Reference . . . . .	1961
5.226.1 Detailed Description . . . . .	1962
5.227losertree.h File Reference . . . . .	1962
5.227.1 Detailed Description . . . . .	1963
5.228lu_counter_metadata.hpp File Reference . . . . .	1963
5.228.1 Detailed Description . . . . .	1963
5.229lu_map_.hpp File Reference . . . . .	1963
5.229.1 Detailed Description . . . . .	1964
5.230macros.h File Reference . . . . .	1964
5.230.1 Detailed Description . . . . .	1964
5.230.2 Macro Definition Documentation . . . . .	1965
5.231malloc_allocator.h File Reference . . . . .	1966
5.231.1 Detailed Description . . . . .	1967
5.232map.h File Reference . . . . .	1967
5.232.1 Detailed Description . . . . .	1967
5.233map.h File Reference . . . . .	1968
5.233.1 Detailed Description . . . . .	1968
5.234mask_array.h File Reference . . . . .	1968
5.234.1 Detailed Description . . . . .	1968
5.235mask_based_range_hashing.hpp File Reference . . . . .	1969
5.235.1 Detailed Description . . . . .	1969
5.236memoryfwd.h File Reference . . . . .	1969
5.236.1 Detailed Description . . . . .	1969
5.237merge.h File Reference . . . . .	1969
5.237.1 Detailed Description . . . . .	1970
5.238messages_members.h File Reference . . . . .	1970
5.238.1 Detailed Description . . . . .	1970
5.239mod_based_range_hashing.hpp File Reference . . . . .	1970
5.239.1 Detailed Description . . . . .	1970

5.240move.h File Reference . . . . .	1970
5.240.1 Detailed Description . . . . .	1971
5.241mt_allocator.h File Reference . . . . .	1971
5.241.1 Detailed Description . . . . .	1972
5.242multimap.h File Reference . . . . .	1972
5.242.1 Detailed Description . . . . .	1972
5.243multimap.h File Reference . . . . .	1973
5.243.1 Detailed Description . . . . .	1973
5.244multisec_selection.h File Reference . . . . .	1973
5.244.1 Detailed Description . . . . .	1974
5.245multiset.h File Reference . . . . .	1974
5.245.1 Detailed Description . . . . .	1975
5.246multiset.h File Reference . . . . .	1975
5.246.1 Detailed Description . . . . .	1975
5.247multiway_merge.h File Reference . . . . .	1975
5.247.1 Detailed Description . . . . .	1978
5.247.2 Macro Definition Documentation . . . . .	1978
5.248multiway_mergesort.h File Reference . . . . .	1978
5.248.1 Detailed Description . . . . .	1979
5.249nested_exception.h File Reference . . . . .	1979
5.249.1 Detailed Description . . . . .	1979
5.250new_allocator.h File Reference . . . . .	1979
5.250.1 Detailed Description . . . . .	1980
5.251node.hpp File Reference . . . . .	1980
5.251.1 Detailed Description . . . . .	1980
5.252node.hpp File Reference . . . . .	1980
5.252.1 Detailed Description . . . . .	1980
5.253node.hpp File Reference . . . . .	1980
5.253.1 Detailed Description . . . . .	1981
5.254node_iterators.hpp File Reference . . . . .	1981
5.254.1 Detailed Description . . . . .	1981
5.255node_iterators.hpp File Reference . . . . .	1981
5.255.1 Detailed Description . . . . .	1981
5.256node_metadata_selector.hpp File Reference . . . . .	1982
5.256.1 Detailed Description . . . . .	1982
5.257node_metadata_selector.hpp File Reference . . . . .	1982
5.257.1 Detailed Description . . . . .	1982

5.258	<a href="#">null_node_metadata.hpp File Reference</a>	1982
5.258.1	Detailed Description	1982
5.259	<a href="#">numeric_traits.h File Reference</a>	1983
5.259.1	Detailed Description	1983
5.260	<a href="#">numeric_fwd.h File Reference</a>	1983
5.260.1	Detailed Description	1984
5.261	<a href="#">omp_loop.h File Reference</a>	1985
5.261.1	Detailed Description	1985
5.262	<a href="#">omp_loop_static.h File Reference</a>	1985
5.262.1	Detailed Description	1985
5.263	<a href="#">opt_random.h File Reference</a>	1985
5.263.1	Detailed Description	1985
5.264	<a href="#">order_statistics_imp.hpp File Reference</a>	1986
5.264.1	Detailed Description	1986
5.265	<a href="#">order_statistics_imp.hpp File Reference</a>	1986
5.265.1	Detailed Description	1986
5.266	<a href="#">os_defines.h File Reference</a>	1986
5.266.1	Detailed Description	1986
5.267	<a href="#">ostream_insert.h File Reference</a>	1986
5.267.1	Detailed Description	1986
5.268	<a href="#">ov_tree_map_.hpp File Reference</a>	1986
5.268.1	Detailed Description	1987
5.269	<a href="#">pairing_heap_.hpp File Reference</a>	1987
5.269.1	Detailed Description	1987
5.270	<a href="#">par_loop.h File Reference</a>	1987
5.270.1	Detailed Description	1988
5.271	<a href="#">parallel.h File Reference</a>	1988
5.271.1	Detailed Description	1988
5.272	<a href="#">partial_sum.h File Reference</a>	1988
5.272.1	Detailed Description	1988
5.273	<a href="#">partition.h File Reference</a>	1988
5.273.1	Detailed Description	1989
5.273.2	Macro Definition Documentation	1989
5.274	<a href="#">pat_trie_.hpp File Reference</a>	1989
5.274.1	Detailed Description	1989
5.275	<a href="#">pat_trie_base.hpp File Reference</a>	1990
5.275.1	Detailed Description	1990

5.276	<a href="#">pod_char_traits.h File Reference</a>	1990
5.276.1	Detailed Description	1991
5.277	<a href="#">point_const_iterator.hpp File Reference</a>	1991
5.277.1	Detailed Description	1991
5.278	<a href="#">point_const_iterator.hpp File Reference</a>	1991
5.278.1	Detailed Description	1991
5.279	<a href="#">point_const_iterator.hpp File Reference</a>	1992
5.279.1	Detailed Description	1992
5.280	<a href="#">point_iterator.hpp File Reference</a>	1992
5.280.1	Detailed Description	1992
5.281	<a href="#">point_iterators.hpp File Reference</a>	1992
5.281.1	Detailed Description	1992
5.282	<a href="#">pointer.h File Reference</a>	1993
5.282.1	Detailed Description	1994
5.283	<a href="#">policy_access_fn_imps.hpp File Reference</a>	1994
5.283.1	Detailed Description	1994
5.284	<a href="#">policy_access_fn_imps.hpp File Reference</a>	1994
5.284.1	Detailed Description	1994
5.285	<a href="#">policy_access_fn_imps.hpp File Reference</a>	1995
5.285.1	Detailed Description	1995
5.286	<a href="#">policy_access_fn_imps.hpp File Reference</a>	1995
5.286.1	Detailed Description	1995
5.287	<a href="#">policy_access_fn_imps.hpp File Reference</a>	1995
5.287.1	Detailed Description	1995
5.288	<a href="#">policy_access_fn_imps.hpp File Reference</a>	1995
5.288.1	Detailed Description	1995
5.289	<a href="#">policy_access_fn_imps.hpp File Reference</a>	1995
5.289.1	Detailed Description	1995
5.290	<a href="#">pool_allocator.h File Reference</a>	1995
5.290.1	Detailed Description	1996
5.291	<a href="#">postypes.h File Reference</a>	1996
5.291.1	Detailed Description	1996
5.292	<a href="#">prefix_search_node_update_imp.hpp File Reference</a>	1996
5.292.1	Detailed Description	1996
5.293	<a href="#">priority_queue.hpp File Reference</a>	1996
5.293.1	Detailed Description	1997
5.294	<a href="#">priority_queue_base_dispatch.hpp File Reference</a>	1997

5.294.1 Detailed Description . . . . .	1997
5.295probe_fn_base.hpp File Reference . . . . .	1997
5.295.1 Detailed Description . . . . .	1997
5.296profiler.h File Reference . . . . .	1997
5.296.1 Detailed Description . . . . .	2000
5.297profiler_algos.h File Reference . . . . .	2000
5.297.1 Detailed Description . . . . .	2000
5.298profiler_container_size.h File Reference . . . . .	2000
5.298.1 Detailed Description . . . . .	2000
5.299profiler_hash_func.h File Reference . . . . .	2001
5.299.1 Detailed Description . . . . .	2001
5.300profiler_hashtable_size.h File Reference . . . . .	2001
5.300.1 Detailed Description . . . . .	2001
5.301profiler_list_to_slist.h File Reference . . . . .	2002
5.301.1 Detailed Description . . . . .	2002
5.302profiler_list_to_vector.h File Reference . . . . .	2002
5.302.1 Detailed Description . . . . .	2002
5.303profiler_map_to_unordered_map.h File Reference . . . . .	2003
5.303.1 Detailed Description . . . . .	2003
5.304profiler_node.h File Reference . . . . .	2003
5.304.1 Detailed Description . . . . .	2004
5.305profiler_state.h File Reference . . . . .	2004
5.305.1 Detailed Description . . . . .	2004
5.306profiler_trace.h File Reference . . . . .	2004
5.306.1 Detailed Description . . . . .	2006
5.307profiler_vector_size.h File Reference . . . . .	2006
5.307.1 Detailed Description . . . . .	2007
5.308profiler_vector_to_list.h File Reference . . . . .	2007
5.308.1 Detailed Description . . . . .	2007
5.309ptr_traits.h File Reference . . . . .	2007
5.309.1 Detailed Description . . . . .	2008
5.310quadratic_probe_fn_imp.hpp File Reference . . . . .	2008
5.310.1 Detailed Description . . . . .	2008
5.311queue.h File Reference . . . . .	2008
5.311.1 Detailed Description . . . . .	2008
5.311.2 Macro Definition Documentation . . . . .	2008
5.312quicksort.h File Reference . . . . .	2008

5.312.1 Detailed Description . . . . .	2009
5.313r_erase_fn_imps.hpp File Reference . . . . .	2009
5.313.1 Detailed Description . . . . .	2009
5.314r_erase_fn_imps.hpp File Reference . . . . .	2009
5.314.1 Detailed Description . . . . .	2009
5.315random.h File Reference . . . . .	2009
5.315.1 Detailed Description . . . . .	2013
5.316random_number.h File Reference . . . . .	2013
5.316.1 Detailed Description . . . . .	2013
5.317random_shuffle.h File Reference . . . . .	2013
5.317.1 Detailed Description . . . . .	2014
5.318range_access.h File Reference . . . . .	2014
5.318.1 Detailed Description . . . . .	2014
5.319ranged_hash_fn.hpp File Reference . . . . .	2014
5.319.1 Detailed Description . . . . .	2015
5.320ranged_probe_fn.hpp File Reference . . . . .	2015
5.320.1 Detailed Description . . . . .	2015
5.321rb_tree_.hpp File Reference . . . . .	2015
5.321.1 Detailed Description . . . . .	2016
5.322rc.hpp File Reference . . . . .	2016
5.322.1 Detailed Description . . . . .	2016
5.323rc_binomial_heap_.hpp File Reference . . . . .	2016
5.323.1 Detailed Description . . . . .	2016
5.324rc_string_base.h File Reference . . . . .	2016
5.324.1 Detailed Description . . . . .	2017
5.325regex.h File Reference . . . . .	2017
5.325.1 Detailed Description . . . . .	2020
5.326regex_compiler.h File Reference . . . . .	2020
5.326.1 Detailed Description . . . . .	2021
5.327regex_constants.h File Reference . . . . .	2021
5.327.1 Detailed Description . . . . .	2022
5.328regex_cursor.h File Reference . . . . .	2022
5.328.1 Detailed Description . . . . .	2022
5.329regex_error.h File Reference . . . . .	2022
5.329.1 Detailed Description . . . . .	2023
5.330regex_grep_matcher.h File Reference . . . . .	2023
5.330.1 Detailed Description . . . . .	2023

5.331	<a href="#">regex_nfa.h File Reference</a>	2024
5.331.1	Detailed Description	2024
5.332	<a href="#">resize_fn_imps.hpp File Reference</a>	2024
5.332.1	Detailed Description	2024
5.333	<a href="#">resize_fn_imps.hpp File Reference</a>	2025
5.333.1	Detailed Description	2025
5.334	<a href="#">resize_no_store_hash_fn_imps.hpp File Reference</a>	2025
5.334.1	Detailed Description	2025
5.335	<a href="#">resize_no_store_hash_fn_imps.hpp File Reference</a>	2025
5.335.1	Detailed Description	2025
5.336	<a href="#">resize_policy.hpp File Reference</a>	2025
5.336.1	Detailed Description	2025
5.337	<a href="#">resize_store_hash_fn_imps.hpp File Reference</a>	2025
5.337.1	Detailed Description	2025
5.338	<a href="#">resize_store_hash_fn_imps.hpp File Reference</a>	2025
5.338.1	Detailed Description	2025
5.339	<a href="#">ropeimpl.h File Reference</a>	2026
5.339.1	Detailed Description	2026
5.340	<a href="#">rotate_fn_imps.hpp File Reference</a>	2026
5.340.1	Detailed Description	2026
5.341	<a href="#">rotate_fn_imps.hpp File Reference</a>	2026
5.341.1	Detailed Description	2026
5.342	<a href="#">safe_base.h File Reference</a>	2026
5.342.1	Detailed Description	2027
5.343	<a href="#">safe_iterator.h File Reference</a>	2027
5.343.1	Detailed Description	2028
5.344	<a href="#">safe_local_iterator.h File Reference</a>	2028
5.344.1	Detailed Description	2028
5.345	<a href="#">safe_sequence.h File Reference</a>	2028
5.345.1	Detailed Description	2029
5.346	<a href="#">safe_unordered_base.h File Reference</a>	2029
5.346.1	Detailed Description	2029
5.347	<a href="#">safe_unordered_container.h File Reference</a>	2029
5.347.1	Detailed Description	2029
5.348	<a href="#">sample_probe_fn.hpp File Reference</a>	2029
5.348.1	Detailed Description	2030
5.349	<a href="#">sample_range_hashing.hpp File Reference</a>	2030

5.349.1 Detailed Description . . . . .	2030
5.350sample_ranged_hash_fn.hpp File Reference . . . . .	2030
5.350.1 Detailed Description . . . . .	2030
5.351sample_ranged_probe_fn.hpp File Reference . . . . .	2030
5.351.1 Detailed Description . . . . .	2031
5.352sample_resize_policy.hpp File Reference . . . . .	2031
5.352.1 Detailed Description . . . . .	2031
5.353sample_resize_trigger.hpp File Reference . . . . .	2031
5.353.1 Detailed Description . . . . .	2031
5.354sample_size_policy.hpp File Reference . . . . .	2031
5.354.1 Detailed Description . . . . .	2031
5.355sample_tree_node_update.hpp File Reference . . . . .	2032
5.355.1 Detailed Description . . . . .	2032
5.356sample_trie_access_traits.hpp File Reference . . . . .	2032
5.356.1 Detailed Description . . . . .	2032
5.357sample_trie_node_update.hpp File Reference . . . . .	2032
5.357.1 Detailed Description . . . . .	2032
5.358sample_update_policy.hpp File Reference . . . . .	2032
5.358.1 Detailed Description . . . . .	2033
5.359search.h File Reference . . . . .	2033
5.359.1 Detailed Description . . . . .	2033
5.360set.h File Reference . . . . .	2033
5.360.1 Detailed Description . . . . .	2034
5.361set.h File Reference . . . . .	2034
5.361.1 Detailed Description . . . . .	2034
5.362set_operations.h File Reference . . . . .	2034
5.362.1 Detailed Description . . . . .	2035
5.363settings.h File Reference . . . . .	2035
5.363.1 Detailed Description . . . . .	2035
5.363.2 parallelization_decision . . . . .	2035
5.363.3 Macro Definition Documentation . . . . .	2036
5.364shared_ptr.h File Reference . . . . .	2036
5.364.1 Detailed Description . . . . .	2037
5.365shared_ptr_base.h File Reference . . . . .	2038
5.365.1 Detailed Description . . . . .	2039
5.366size_fn_imps.hpp File Reference . . . . .	2039
5.366.1 Detailed Description . . . . .	2039



5.367	<a href="#">slice_array.h File Reference</a>	2039
5.367.1	<a href="#">Detailed Description</a>	2040
5.368	<a href="#">sort.h File Reference</a>	2040
5.368.1	<a href="#">Detailed Description</a>	2040
5.369	<a href="#">splay_fn_imps.hpp File Reference</a>	2040
5.369.1	<a href="#">Detailed Description</a>	2040
5.370	<a href="#">splay_tree_.hpp File Reference</a>	2040
5.370.1	<a href="#">Detailed Description</a>	2041
5.371	<a href="#">split_fn_imps.hpp File Reference</a>	2041
5.371.1	<a href="#">Detailed Description</a>	2041
5.372	<a href="#">split_join_fn_imps.hpp File Reference</a>	2041
5.372.1	<a href="#">Detailed Description</a>	2041
5.373	<a href="#">split_join_fn_imps.hpp File Reference</a>	2041
5.373.1	<a href="#">Detailed Description</a>	2041
5.374	<a href="#">split_join_fn_imps.hpp File Reference</a>	2041
5.374.1	<a href="#">Detailed Description</a>	2041
5.375	<a href="#">split_join_fn_imps.hpp File Reference</a>	2041
5.375.1	<a href="#">Detailed Description</a>	2041
5.376	<a href="#">split_join_fn_imps.hpp File Reference</a>	2041
5.376.1	<a href="#">Detailed Description</a>	2041
5.377	<a href="#">split_join_fn_imps.hpp File Reference</a>	2042
5.377.1	<a href="#">Detailed Description</a>	2042
5.378	<a href="#">split_join_fn_imps.hpp File Reference</a>	2042
5.378.1	<a href="#">Detailed Description</a>	2042
5.379	<a href="#">split_join_fn_imps.hpp File Reference</a>	2042
5.379.1	<a href="#">Detailed Description</a>	2042
5.380	<a href="#">split_join_fn_imps.hpp File Reference</a>	2042
5.380.1	<a href="#">Detailed Description</a>	2042
5.381	<a href="#">sso_string_base.h File Reference</a>	2042
5.381.1	<a href="#">Detailed Description</a>	2042
5.382	<a href="#">standard_policies.hpp File Reference</a>	2042
5.382.1	<a href="#">Detailed Description</a>	2043
5.383	<a href="#">stdc++.h File Reference</a>	2043
5.383.1	<a href="#">Detailed Description</a>	2043
5.384	<a href="#">stdio_filebuf.h File Reference</a>	2043
5.384.1	<a href="#">Detailed Description</a>	2043
5.385	<a href="#">stdio_sync_filebuf.h File Reference</a>	2043

5.385.1 Detailed Description . . . . .	2044
5.386stdtr1c++.h File Reference . . . . .	2044
5.386.1 Detailed Description . . . . .	2044
5.387stl_algo.h File Reference . . . . .	2044
5.387.1 Detailed Description . . . . .	2052
5.388stl_algobase.h File Reference . . . . .	2052
5.388.1 Detailed Description . . . . .	2053
5.389stl_bvector.h File Reference . . . . .	2054
5.389.1 Detailed Description . . . . .	2054
5.390stl_construct.h File Reference . . . . .	2054
5.390.1 Detailed Description . . . . .	2055
5.391stl_deque.h File Reference . . . . .	2055
5.391.1 Detailed Description . . . . .	2056
5.391.2 Macro Definition Documentation . . . . .	2056
5.392stl_function.h File Reference . . . . .	2057
5.392.1 Detailed Description . . . . .	2058
5.393stl_heap.h File Reference . . . . .	2058
5.393.1 Detailed Description . . . . .	2059
5.394stl_iterator.h File Reference . . . . .	2059
5.394.1 Detailed Description . . . . .	2062
5.395stl_iterator_base_funcs.h File Reference . . . . .	2062
5.395.1 Detailed Description . . . . .	2062
5.396stl_iterator_base_types.h File Reference . . . . .	2063
5.396.1 Detailed Description . . . . .	2063
5.397stl_list.h File Reference . . . . .	2063
5.397.1 Detailed Description . . . . .	2064
5.398stl_map.h File Reference . . . . .	2064
5.398.1 Detailed Description . . . . .	2065
5.399stl_multimap.h File Reference . . . . .	2065
5.399.1 Detailed Description . . . . .	2065
5.400stl_multiset.h File Reference . . . . .	2065
5.400.1 Detailed Description . . . . .	2066
5.401stl_numeric.h File Reference . . . . .	2066
5.401.1 Detailed Description . . . . .	2067
5.402stl_pair.h File Reference . . . . .	2067
5.402.1 Detailed Description . . . . .	2067
5.403stl_queue.h File Reference . . . . .	2067

5.403.1 Detailed Description . . . . .	2068
5.404stl_raw_storage_iter.h File Reference . . . . .	2068
5.404.1 Detailed Description . . . . .	2068
5.405stl_relops.h File Reference . . . . .	2068
5.405.1 Detailed Description . . . . .	2069
5.406stl_set.h File Reference . . . . .	2069
5.406.1 Detailed Description . . . . .	2069
5.407stl_stack.h File Reference . . . . .	2070
5.407.1 Detailed Description . . . . .	2070
5.408stl_tempbuf.h File Reference . . . . .	2070
5.408.1 Detailed Description . . . . .	2071
5.409stl_tree.h File Reference . . . . .	2071
5.409.1 Detailed Description . . . . .	2072
5.410stl_uninitialized.h File Reference . . . . .	2072
5.410.1 Detailed Description . . . . .	2073
5.411stl_vector.h File Reference . . . . .	2073
5.411.1 Detailed Description . . . . .	2074
5.412stream_iterator.h File Reference . . . . .	2074
5.412.1 Detailed Description . . . . .	2074
5.413streambuf_iterator.h File Reference . . . . .	2074
5.413.1 Detailed Description . . . . .	2075
5.414string_conversions.h File Reference . . . . .	2075
5.414.1 Detailed Description . . . . .	2075
5.415stringfwd.h File Reference . . . . .	2075
5.415.1 Detailed Description . . . . .	2076
5.416synth_access_traits.hpp File Reference . . . . .	2076
5.416.1 Detailed Description . . . . .	2076
5.417tag_and_trait.hpp File Reference . . . . .	2076
5.417.1 Detailed Description . . . . .	2077
5.418tags.h File Reference . . . . .	2077
5.418.1 Detailed Description . . . . .	2078
5.419tgmath.h File Reference . . . . .	2078
5.419.1 Detailed Description . . . . .	2078
5.420thin_heap_.hpp File Reference . . . . .	2078
5.420.1 Detailed Description . . . . .	2079
5.421throw_allocator.h File Reference . . . . .	2079
5.421.1 Detailed Description . . . . .	2080

5.422time_members.h File Reference . . . . .	2080
5.422.1 Detailed Description . . . . .	2080
5.423trace_fn_imps.hpp File Reference . . . . .	2080
5.423.1 Detailed Description . . . . .	2080
5.424trace_fn_imps.hpp File Reference . . . . .	2080
5.424.1 Detailed Description . . . . .	2080
5.425trace_fn_imps.hpp File Reference . . . . .	2081
5.425.1 Detailed Description . . . . .	2081
5.426trace_fn_imps.hpp File Reference . . . . .	2081
5.426.1 Detailed Description . . . . .	2081
5.427trace_fn_imps.hpp File Reference . . . . .	2081
5.427.1 Detailed Description . . . . .	2081
5.428trace_fn_imps.hpp File Reference . . . . .	2081
5.428.1 Detailed Description . . . . .	2081
5.429trace_fn_imps.hpp File Reference . . . . .	2081
5.429.1 Detailed Description . . . . .	2081
5.430trace_fn_imps.hpp File Reference . . . . .	2081
5.430.1 Detailed Description . . . . .	2081
5.431traits.hpp File Reference . . . . .	2081
5.431.1 Detailed Description . . . . .	2082
5.432traits.hpp File Reference . . . . .	2082
5.432.1 Detailed Description . . . . .	2082
5.433traits.hpp File Reference . . . . .	2082
5.433.1 Detailed Description . . . . .	2082
5.434traits.hpp File Reference . . . . .	2082
5.434.1 Detailed Description . . . . .	2083
5.435traits.hpp File Reference . . . . .	2083
5.435.1 Detailed Description . . . . .	2083
5.436traits.hpp File Reference . . . . .	2083
5.436.1 Detailed Description . . . . .	2083
5.437tree_policy.hpp File Reference . . . . .	2083
5.437.1 Detailed Description . . . . .	2084
5.438tree_trace_base.hpp File Reference . . . . .	2084
5.438.1 Detailed Description . . . . .	2084
5.439trie_policy.hpp File Reference . . . . .	2084
5.439.1 Detailed Description . . . . .	2084
5.440trie_policy_base.hpp File Reference . . . . .	2084

5.440.1 Detailed Description . . . . .	2085
5.441trie_string_access_traits_imp.hpp File Reference . . . . .	2085
5.441.1 Detailed Description . . . . .	2085
5.442type_traits.h File Reference . . . . .	2085
5.442.1 Detailed Description . . . . .	2085
5.443type_utils.hpp File Reference . . . . .	2085
5.443.1 Detailed Description . . . . .	2086
5.444typelist.h File Reference . . . . .	2086
5.444.1 Detailed Description . . . . .	2087
5.445types.h File Reference . . . . .	2087
5.445.1 Detailed Description . . . . .	2087
5.446types_traits.hpp File Reference . . . . .	2087
5.446.1 Detailed Description . . . . .	2088
5.447unique_copy.h File Reference . . . . .	2088
5.447.1 Detailed Description . . . . .	2088
5.448unique_ptr.h File Reference . . . . .	2088
5.448.1 Detailed Description . . . . .	2089
5.449unordered_base.h File Reference . . . . .	2089
5.449.1 Detailed Description . . . . .	2090
5.450unordered_map.h File Reference . . . . .	2090
5.450.1 Detailed Description . . . . .	2090
5.451unordered_set.h File Reference . . . . .	2091
5.451.1 Detailed Description . . . . .	2091
5.452update_fn_imps.hpp File Reference . . . . .	2091
5.452.1 Detailed Description . . . . .	2091
5.453valarray_after.h File Reference . . . . .	2092
5.453.1 Detailed Description . . . . .	2099
5.454valarray_array.h File Reference . . . . .	2099
5.454.1 Detailed Description . . . . .	2105
5.455valarray_before.h File Reference . . . . .	2105
5.455.1 Detailed Description . . . . .	2105
5.456vstring.h File Reference . . . . .	2105
5.456.1 Detailed Description . . . . .	2107
5.457vstring_fwd.h File Reference . . . . .	2107
5.457.1 Detailed Description . . . . .	2108
5.458vstring_util.h File Reference . . . . .	2108
5.458.1 Detailed Description . . . . .	2108

5.459workstealing.h File Reference . . . . .	2108
5.459.1 Detailed Description . . . . .	2109
<b>Index</b>	<b>2111</b>

## 1 Todo List

### globalScope> Member [\\_\\_glibcxx\\_check\\_insert\\_range](#) ([\\_Position](#), [\\_First](#), [\\_Last](#))

We would like to be able to check for noninterference of [\\_Position](#) and the range [[\\_First](#), [\\_Last](#)), but that can't (in general) be done.

### globalScope> Member [\\_\\_glibcxx\\_check\\_insert\\_range\\_after](#) ([\\_Position](#), [\\_First](#), [\\_Last](#))

We would like to be able to check for noninterference of [\\_Position](#) and the range [[\\_First](#), [\\_Last](#)), but that can't (in general) be done.

### Member [\\_\\_gnu\\_debug::Safe\\_iterator](#)< [\\_Iterator](#), [\\_Sequence](#) >::operator-> () const

Make this correct w.r.t. iterators that return proxies

Use addressof() instead of & operator

### Member [\\_\\_gnu\\_debug::Safe\\_local\\_iterator](#)< [\\_Iterator](#), [\\_Sequence](#) >::operator-> () const

Make this correct w.r.t. iterators that return proxies

Use addressof() instead of & operator

### Class [std::basic\\_string](#)< [\\_CharT](#), [\\_Traits](#), [\\_Alloc](#) >

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

### Member [std::match\\_results](#)< [\\_Bi\\_iter](#), [\\_Alloc](#) >::format ([\\_Out\\_iter](#) [\\_\\_out](#), const [char\\_type](#) \* [\\_\\_fmt\\_first](#), const [char\\_type](#) \* [\\_\\_fmt\\_last](#), [match\\_flag\\_type](#) [\\_\\_flags](#)=[regex\\_constants::format\\_default](#)) const

Implement this function.

### Member [std::regex\\_iterator](#)< [\\_Bi\\_iter](#), [\\_Ch\\_type](#), [\\_Rx\\_traits](#) >::operator!= (const [regex\\_iterator](#) & [\\_\\_rhs](#))

Implement this function.

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

### Member [std::regex\\_iterator](#)< [\\_Bi\\_iter](#), [\\_Ch\\_type](#), [\\_Rx\\_traits](#) >::operator\* ()

Implement this function.

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

### Member [std::regex\\_iterator](#)< [\\_Bi\\_iter](#), [\\_Ch\\_type](#), [\\_Rx\\_traits](#) >::operator++ ()

Implement this function.

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

### Member [std::regex\\_iterator](#)< [\\_Bi\\_iter](#), [\\_Ch\\_type](#), [\\_Rx\\_traits](#) >::operator++ (int)

Implement this function.

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

Member **std::regex\_iterator<\_Bi\_iter, \_Ch\_type, \_Rx\_traits>::operator->()**

Implement this function.

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

Member **std::regex\_iterator<\_Bi\_iter, \_Ch\_type, \_Rx\_traits>::operator= (const regex\_iterator &\_\_rhs)**

Implement this function.

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

Member **std::regex\_iterator<\_Bi\_iter, \_Ch\_type, \_Rx\_traits>::operator== (const regex\_iterator &\_\_rhs)**

Implement this function.

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

Member **std::regex\_iterator<\_Bi\_iter, \_Ch\_type, \_Rx\_traits>::regex\_iterator ()**

Implement this function.

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

Member **std::regex\_iterator<\_Bi\_iter, \_Ch\_type, \_Rx\_traits>::regex\_iterator (\_Bi\_iter \_\_a, \_Bi\_iter \_\_b, const regex\_type &\_\_re, regex\_constants::match\_flag\_type \_\_m=regex\_constants::match\_default)**

Implement this function.

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

Member **std::regex\_iterator<\_Bi\_iter, \_Ch\_type, \_Rx\_traits>::regex\_iterator (const regex\_iterator &\_\_rhs)**

Implement this function.

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

Member **std::regex\_match (\_Bi\_iter \_\_s, \_Bi\_iter \_\_e, match\_results<\_Bi\_iter, \_Alloc> &\_\_m, const basic\_↵\_regex<\_Ch\_type, \_Rx\_traits> &\_\_re, regex\_constants::match\_flag\_type \_\_flags=regex\_constants\_↵::match\_default)**

Implement this function.

Member **std::regex\_replace (\_Out\_iter \_\_out, \_Bi\_iter \_\_first, \_Bi\_iter \_\_last, const basic\_regex<\_Ch\_↵type, \_Rx\_traits> &\_\_e, const basic\_string<\_Ch\_type> &\_\_fmt, regex\_constants::match\_flag\_type \_↵\_\_flags=regex\_constants::match\_default)**

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

Parameters

<i>__out</i>	
<i>__first</i>	
<i>__last</i>	
<i>__e</i>	
<i>__fmt</i>	
<i>__flags&lt;p&gt;↵ Implement</i>	this function.

Implement this function.

Member `std::regex_replace` (const basic\_string< \_Ch\_type > &\_\_s, const basic\_regex< \_Ch\_type, \_Rx\_traits > &\_\_e, const basic\_string< \_Ch\_type > &\_\_fmt, regex\_constants::match\_flag\_type \_\_flags=regex\_constants::match\_default)

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



## Parameters

<code>__s</code>	
<code>__e</code>	
<code>__fmt</code>	
<code>__flags</code>	

Member **`std::regex_search`** (`const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s`, `const basic_regex< _Ch_type, _Rx_traits > &__e`, `regex_constants::match_flag_type __flags=regex_constants::match_default`)

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

Member **`std::regex_search`** (`_Bi_iter __first`, `_Bi_iter __last`, `const basic_regex< _Ch_type, _Rx_traits > &__re`, `regex_constants::match_flag_type __flags=regex_constants::match_default`)

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

Member **`std::regex_search`** (`const _Ch_type *__s`, `const basic_regex< _Ch_type, _Rx_traits > &__e`, `regex_constants::match_flag_type __f=regex_constants::match_default`)

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

Member **`std::regex_search`** (`_Bi_iter __first`, `_Bi_iter __last`, `match_results< _Bi_iter, _Alloc > &__m`, `const basic_regex< _Ch_type, _Rx_traits > &__re`, `regex_constants::match_flag_type __flags=regex_constants::match_default`)

Implement this function.

Member **`std::regex_search`** (`const _Ch_type *__s`, `match_results< const _Ch_type *, _Alloc > &__m`, `const basic_regex< _Ch_type, _Rx_traits > &__e`, `regex_constants::match_flag_type __f=regex_constants::match_default`)

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

Member **`std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator!=`** (`const regex_token_iterator &__rhs`)

Implement this function.

Member **`std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator*()`**

Implement this function.

Member **`std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++()`**

Implement this function.

Member **`std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++(int)`**

Implement this function.

Member **`std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator->()`**

Implement this function.

Member **`std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator=`** (`const regex_token_iterator &__rhs`)

Implement this function.

Member **`std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator==`** (`const regex_token_iterator &__rhs`)

Implement this function.

Member **`std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator()`**

Implement this function.

Member `std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_token_iterator` (`_Bi_iter __a, _Bi_iter __b, const regex_type &__re, const std::vector<int> &__submatches, regex_constants::match_flag_type __m=regex_constants::match_default`)

Implement this function.

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

Member `std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_token_iterator` (`const regex_token_iterator &__rhs`)

Implement this function.

Member `std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_token_iterator` (`_Bi_iter __a, _Bi_iter __b, const regex_type &__re, const int(&__submatches)[Nm], regex_constants::match_flag_type __m=regex_constants::match_default`)

Implement this function.

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

Member `std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_token_iterator` (`_Bi_iter __a, _Bi_iter __b, const regex_type &__re, int __submatch=0, regex_constants::match_flag_type __m=regex_constants::match_default`)

Implement this function.

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

Member `std::regex_traits<_Ch_type>::lookup_classname` (`_Fwd_iter __first, _Fwd_iter __last, bool __icase=false`) const

Implement this function.

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

Implement this function.

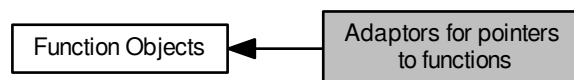
Member `std::regex_traits<_Ch_type>::transform_primary` (`_Fwd_iter __first, _Fwd_iter __last`) const

Implement this function.

## 2 Module Documentation

### 2.1 Adaptors for pointers to functions

Collaboration diagram for Adaptors for pointers to functions:



## Classes

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

## Functions

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

### 2.1.1 Detailed Description

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

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

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

### 2.1.2 Function Documentation

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

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

Definition at line 442 of file `stl_function.h`.

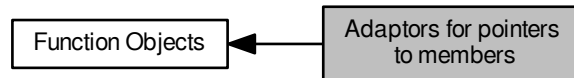
**2.1.2.2** `template<typename _Arg1, typename _Arg2, typename _Result > pointer_to_binary_function< _Arg1, _Arg2, _Result > std::ptr_fun ( _Result(*)(_Arg1, _Arg2) __x ) [inline]`

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

Definition at line 468 of file `stl_function.h`.

## 2.2 Adaptors for pointers to members

Collaboration diagram for Adaptors for pointers to members:



### Classes

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

### Functions

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

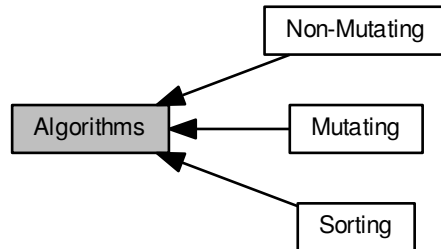
#### 2.2.1 Detailed Description

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

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

## 2.3 Algorithms

Collaboration diagram for Algorithms:



### Modules

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

### 2.3.1 Detailed Description

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

## 2.4 Allocators

### Classes

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

### Typedefs

- template<typename `_Tp`>  
using `std::__allocator_base` = `__gnu_cxx::new_allocator<_Tp>`

### Enumerations

- enum { **modulus**, **value** }

### Functions

- template<typename `_T1`, typename `_T2`>  
bool **std::operator!=** (const allocator< `_T1` > &, const allocator< `_T2` > &)
- template<typename `_Tp`>  
bool **std::operator!=** (const allocator< `_Tp` > &, const allocator< `_Tp` > &)
- template<typename `_T1`, typename `_T2`>  
bool **std::operator==** (const allocator< `_T1` > &, const allocator< `_T2` > &)
- template<typename `_Tp`>  
bool **std::operator==** (const allocator< `_Tp` > &, const allocator< `_Tp` > &)

#### 2.4.1 Detailed Description

Classes encapsulating memory operations.

#### 2.4.2 Typedef Documentation

##### 2.4.2.1 `template<typename _Tp> using std::__allocator_base = typedef __gnu_cxx::new_allocator<_Tp>`

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

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

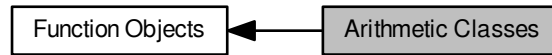
**Template Parameters**

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

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

## 2.5 Arithmetic Classes

Collaboration diagram for Arithmetic Classes:



### Classes

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

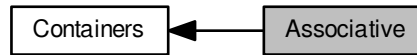
### 2.5.1 Detailed Description

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



## 2.6 Associative

Collaboration diagram for Associative:



### Classes

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

### 2.6.1 Detailed Description

Associative containers allow fast retrieval of data based on keys.

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

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

## 2.7 Atomics

### Classes

- struct `std::__atomic_base<_ITp>`
- struct `std::__atomic_base<_PTp*>`
- struct `std::__atomic_flag_base`
- struct `std::atomic_flag`

### Macros

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

### Typedefs

- typedef unsigned char `std::__atomic_flag_data_type`
- typedef `__atomic_base<char>` `std::atomic_char`
- typedef `__atomic_base<char16_t>` `std::atomic_char16_t`
- typedef `__atomic_base<char32_t>` `std::atomic_char32_t`
- typedef `__atomic_base<int>` `std::atomic_int`
- typedef `__atomic_base<int_fast16_t>` `std::atomic_int_fast16_t`
- typedef `__atomic_base<int_fast32_t>` `std::atomic_int_fast32_t`
- typedef `__atomic_base<int_fast64_t>` `std::atomic_int_fast64_t`
- typedef `__atomic_base<int_fast8_t>` `std::atomic_int_fast8_t`
- typedef `__atomic_base<int_least16_t>` `std::atomic_int_least16_t`
- typedef `__atomic_base<int_least32_t>` `std::atomic_int_least32_t`
- typedef `__atomic_base<int_least64_t>` `std::atomic_int_least64_t`
- typedef `__atomic_base<int_least8_t>` `std::atomic_int_least8_t`
- typedef `__atomic_base<intmax_t>` `std::atomic_intmax_t`
- typedef `__atomic_base<intptr_t>` `std::atomic_intptr_t`
- typedef `__atomic_base<long long>` `std::atomic_llong`
- typedef `__atomic_base<long>` `std::atomic_long`
- typedef `__atomic_base<ptrdiff_t>` `std::atomic_ptrdiff_t`
- typedef `__atomic_base<signed char>` `std::atomic_schar`
- typedef `__atomic_base<short>` `std::atomic_short`
- typedef `__atomic_base<size_t>` `std::atomic_size_t`
- typedef `__atomic_base<unsigned char>` `std::atomic_uchar`
- typedef `__atomic_base<unsigned int>` `std::atomic_uint`
- typedef `__atomic_base<uint_fast16_t>` `std::atomic_uint_fast16_t`

- typedef \_\_atomic\_base< uint\_fast32\_t > [std::atomic\\_uint\\_fast32\\_t](#)
- typedef \_\_atomic\_base< uint\_fast64\_t > [std::atomic\\_uint\\_fast64\\_t](#)
- typedef \_\_atomic\_base< uint\_fast8\_t > [std::atomic\\_uint\\_fast8\\_t](#)
- typedef \_\_atomic\_base< uint\_least16\_t > [std::atomic\\_uint\\_least16\\_t](#)
- typedef \_\_atomic\_base< uint\_least32\_t > [std::atomic\\_uint\\_least32\\_t](#)
- typedef \_\_atomic\_base< uint\_least64\_t > [std::atomic\\_uint\\_least64\\_t](#)
- typedef \_\_atomic\_base< uint\_least8\_t > [std::atomic\\_uint\\_least8\\_t](#)
- typedef \_\_atomic\_base< uintmax\_t > [std::atomic\\_uintmax\\_t](#)
- typedef \_\_atomic\_base< uintptr\_t > [std::atomic\\_uintptr\\_t](#)
- typedef \_\_atomic\_base< unsigned long long > [std::atomic\\_ullong](#)
- typedef \_\_atomic\_base< unsigned long > [std::atomic\\_ulong](#)
- typedef \_\_atomic\_base< unsigned short > [std::atomic\\_ushort](#)
- typedef \_\_atomic\_base< wchar\_t > [std::atomic\\_wchar\\_t](#)
- typedef enum [std::memory\\_order](#) [std::memory\\_order](#)

## Enumerations

- enum \_\_memory\_order\_modifier { \_\_memory\_order\_mask, \_\_memory\_order\_modifier\_mask, \_\_memory\_order\_hle\_acquire, \_\_memory\_order\_hle\_release }
- enum [std::memory\\_order](#) { [memory\\_order\\_relaxed](#), [memory\\_order\\_consume](#), [memory\\_order\\_acquire](#), [memory\\_order\\_release](#), [memory\\_order\\_acq\\_rel](#), [memory\\_order\\_seq\\_cst](#) }

## Functions

- constexpr memory\_order [std::\\_\\_cmpexch\\_failure\\_order](#) (memory\_order \_\_m) noexcept
- constexpr memory\_order [std::\\_\\_cmpexch\\_failure\\_order2](#) (memory\_order \_\_m) noexcept
- void [std::atomic\\_signal\\_fence](#) (memory\_order \_\_m) noexcept
- void [std::atomic\\_thread\\_fence](#) (memory\_order \_\_m) noexcept
- template<typename \_Tp >  
\_Tp [std::kill\\_dependency](#) (\_Tp \_\_y) noexcept
- constexpr memory\_order [std::operator&](#) (memory\_order \_\_m, \_\_memory\_order\_modifier \_\_mod)
- constexpr memory\_order [std::operator|](#) (memory\_order \_\_m, \_\_memory\_order\_modifier \_\_mod)

### 2.7.1 Detailed Description

Components for performing atomic operations.

### 2.7.2 Macro Definition Documentation

#### 2.7.2.1 #define ATOMIC\_BOOL\_LOCK\_FREE

Lock-free property.

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

Definition at line 49 of file [atomic\\_lockfree\\_defines.h](#).

### 2.7.3 Typedef Documentation

#### 2.7.3.1 typedef \_\_atomic\_base<char> std::atomic\_char

atomic\_char

Definition at line 117 of file atomic\_base.h.

#### 2.7.3.2 typedef \_\_atomic\_base<char16\_t> std::atomic\_char16\_t

atomic\_char16\_t

Definition at line 156 of file atomic\_base.h.

#### 2.7.3.3 typedef \_\_atomic\_base< char32\_t > std::atomic\_char32\_t

atomic\_char32\_t

Definition at line 159 of file atomic\_base.h.

#### 2.7.3.4 typedef \_\_atomic\_base<int> std::atomic\_int

atomic\_int

Definition at line 135 of file atomic\_base.h.

#### 2.7.3.5 typedef \_\_atomic\_base<int\_fast16\_t> std::atomic\_int\_fast16\_t

atomic\_int\_fast16\_t

Definition at line 197 of file atomic\_base.h.

#### 2.7.3.6 typedef \_\_atomic\_base<int\_fast32\_t> std::atomic\_int\_fast32\_t

atomic\_int\_fast32\_t

Definition at line 203 of file atomic\_base.h.

#### 2.7.3.7 typedef \_\_atomic\_base<int\_fast64\_t> std::atomic\_int\_fast64\_t

atomic\_int\_fast64\_t

Definition at line 209 of file atomic\_base.h.

#### 2.7.3.8 typedef \_\_atomic\_base<int\_fast8\_t> std::atomic\_int\_fast8\_t

atomic\_int\_fast8\_t

Definition at line 191 of file atomic\_base.h.

#### 2.7.3.9 typedef \_\_atomic\_base<int\_least16\_t> std::atomic\_int\_least16\_t

atomic\_int\_least16\_t

Definition at line 172 of file atomic\_base.h.

#### 2.7.3.10 typedef \_\_atomic\_base<int\_least32\_t> std::atomic\_int\_least32\_t

atomic\_int\_least32\_t

Definition at line 178 of file atomic\_base.h.

**2.7.3.11** `typedef __atomic_base<int_least64_t> std::atomic_int_least64_t`

`atomic_int_least64_t`

Definition at line 184 of file `atomic_base.h`.

**2.7.3.12** `typedef __atomic_base<int_least8_t> std::atomic_int_least8_t`

`atomic_int_least8_t`

Definition at line 166 of file `atomic_base.h`.

**2.7.3.13** `typedef __atomic_base<intmax_t> std::atomic_intmax_t`

`atomic_intmax_t`

Definition at line 225 of file `atomic_base.h`.

**2.7.3.14** `typedef __atomic_base<intptr_t> std::atomic_intptr_t`

`atomic_intptr_t`

Definition at line 216 of file `atomic_base.h`.

**2.7.3.15** `typedef __atomic_base<long long> std::atomic_llong`

`atomic_llong`

Definition at line 147 of file `atomic_base.h`.

**2.7.3.16** `typedef __atomic_base<long> std::atomic_long`

`atomic_long`

Definition at line 141 of file `atomic_base.h`.

**2.7.3.17** `typedef __atomic_base<ptrdiff_t> std::atomic_ptrdiff_t`

`atomic_ptrdiff_t`

Definition at line 231 of file `atomic_base.h`.

**2.7.3.18** `typedef __atomic_base<signed char> std::atomic_schar`

`atomic_schar`

Definition at line 123 of file `atomic_base.h`.

**2.7.3.19** `typedef __atomic_base<short> std::atomic_short`

`atomic_short`

Definition at line 129 of file `atomic_base.h`.

**2.7.3.20** `typedef __atomic_base<size_t> std::atomic_size_t`

`atomic_size_t`

Definition at line 222 of file `atomic_base.h`.

**2.7.3.21** `typedef __atomic_base<unsigned char> std::atomic_uchar`

`atomic_uchar`

Definition at line 126 of file `atomic_base.h`.

**2.7.3.22** `typedef __atomic_base<unsigned int> std::atomic_uint`

`atomic_uint`

Definition at line 138 of file `atomic_base.h`.

**2.7.3.23** `typedef __atomic_base<uint_fast16_t> std::atomic_uint_fast16_t`

`atomic_uint_fast16_t`

Definition at line 200 of file `atomic_base.h`.

**2.7.3.24** `typedef __atomic_base<uint_fast32_t> std::atomic_uint_fast32_t`

`atomic_uint_fast32_t`

Definition at line 206 of file `atomic_base.h`.

**2.7.3.25** `typedef __atomic_base<uint_fast64_t> std::atomic_uint_fast64_t`

`atomic_uint_fast64_t`

Definition at line 212 of file `atomic_base.h`.

**2.7.3.26** `typedef __atomic_base<uint_fast8_t> std::atomic_uint_fast8_t`

`atomic_uint_fast8_t`

Definition at line 194 of file `atomic_base.h`.

**2.7.3.27** `typedef __atomic_base<uint_least16_t> std::atomic_uint_least16_t`

`atomic_uint_least16_t`

Definition at line 175 of file `atomic_base.h`.

**2.7.3.28** `typedef __atomic_base<uint_least32_t> std::atomic_uint_least32_t`

`atomic_uint_least32_t`

Definition at line 181 of file `atomic_base.h`.

**2.7.3.29** `typedef __atomic_base<uint_least64_t> std::atomic_uint_least64_t`

`atomic_uint_least64_t`

Definition at line 187 of file `atomic_base.h`.

**2.7.3.30** `typedef __atomic_base<uint_least8_t> std::atomic_uint_least8_t`

`atomic_uint_least8_t`

Definition at line 169 of file `atomic_base.h`.

### 2.7.3.31 `typedef __atomic_base<uintmax_t> std::atomic_uintmax_t`

`atomic_uintmax_t`

Definition at line 228 of file `atomic_base.h`.

### 2.7.3.32 `typedef __atomic_base<uintptr_t> std::atomic_uintptr_t`

`atomic_uintptr_t`

Definition at line 219 of file `atomic_base.h`.

### 2.7.3.33 `typedef __atomic_base<unsigned long long> std::atomic_ullong`

`atomic_ullong`

Definition at line 150 of file `atomic_base.h`.

### 2.7.3.34 `typedef __atomic_base<unsigned long> std::atomic_ulong`

`atomic_ulong`

Definition at line 144 of file `atomic_base.h`.

### 2.7.3.35 `typedef __atomic_base<unsigned short> std::atomic_ushort`

`atomic_ushort`

Definition at line 132 of file `atomic_base.h`.

### 2.7.3.36 `typedef __atomic_base<wchar_t> std::atomic_wchar_t`

`atomic_wchar_t`

Definition at line 153 of file `atomic_base.h`.

### 2.7.3.37 `typedef enum std::memory_order std::memory_order`

Enumeration for `memory_order`.

## 2.7.4 Enumeration Type Documentation

### 2.7.4.1 `enum std::memory_order`

Enumeration for `memory_order`.

Definition at line 52 of file `atomic_base.h`.

## 2.7.5 Function Documentation

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

`kill_dependency`

Definition at line 108 of file `atomic_base.h`.

## 2.8 Base and Implementation Classes

Collaboration diagram for Base and Implementation Classes:



### Classes

- struct std::\_\_detail::\_Before\_begin< \_NodeAlloc >
- struct std::\_\_detail::\_Default\_ranged\_hash
- struct std::\_\_detail::\_Equal\_helper< \_Key, \_Value, \_ExtractKey, \_Equal, \_HashCodeType, \_\_cache\_hash\_code >
- struct std::\_\_detail::\_Equal\_helper< \_Key, \_Value, \_ExtractKey, \_Equal, \_HashCodeType, false >
- struct std::\_\_detail::\_Equal\_helper< \_Key, \_Value, \_ExtractKey, \_Equal, \_HashCodeType, true >
- struct std::\_\_detail::\_Equality< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, \_Unique\_keys >
- struct std::\_\_detail::\_Equality< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, false >
- struct std::\_\_detail::\_Equality< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, true >
- struct std::\_\_detail::\_Equality\_base
- struct std::\_\_detail::\_Hash\_code\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_cache\_hash\_code >
- struct std::\_\_detail::\_Hash\_code\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Default\_ranged\_hash, false >
- struct std::\_\_detail::\_Hash\_code\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Default\_ranged\_hash, true >
- struct std::\_\_detail::\_Hash\_code\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, false >
- struct std::\_\_detail::\_Hash\_node< \_Value, \_Cache\_hash\_code >
- struct std::\_\_detail::\_Hash\_node< \_Value, false >
- struct std::\_\_detail::\_Hash\_node< \_Value, true >
- struct std::\_\_detail::\_Hash\_node\_base
- struct std::\_\_detail::\_Hashtable\_base< \_Key, \_Value, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_Traits >
- struct std::\_\_detail::\_Hashtable\_ebo\_helper< \_Nm, \_Tp, \_\_use\_ebo >
- struct std::\_\_detail::\_Hashtable\_ebo\_helper< \_Nm, \_Tp, false >
- struct std::\_\_detail::\_Hashtable\_ebo\_helper< \_Nm, \_Tp, true >
- struct std::\_\_detail::\_Hashtable\_traits< \_Cache\_hash\_code, \_Constant\_iterators, \_Unique\_keys >
- struct std::\_\_detail::\_Insert< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_\_ Traits, \_Constant\_iterators, \_Unique\_keys >
- struct std::\_\_detail::\_Insert< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_\_ Traits, false, \_Unique\_keys >
- struct std::\_\_detail::\_Insert< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_\_ Traits, true, false >
- struct std::\_\_detail::\_Insert< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_\_ Traits, true, true >



- struct `std::__detail::Insert_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`
- struct `std::__detail::Local_const_iterator<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >`
- struct `std::__detail::Local_iterator<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >`
- struct `std::__detail::Local_iterator_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >`
- struct `std::__detail::Local_iterator_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >`
- struct `std::__detail::Local_iterator_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >`
- struct `std::__detail::Map_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >`
- struct `std::__detail::Map_base<_Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, __Traits, false >`
- struct `std::__detail::Map_base<_Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, __Traits, true >`
- struct `std::__detail::Mod_range_hashing`
- struct `std::__detail::Node_const_iterator<_Value, __constant_iterators, __cache >`
- struct `std::__detail::Node_iterator<_Value, __constant_iterators, __cache >`
- struct `std::__detail::Node_iterator_base<_Value, _Cache_hash_code >`
- struct `std::__detail::Prime_rehash_policy`
- struct `std::__detail::Rehash_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`
- struct `std::__detail::Rehash_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _Prime_rehash_policy, _Traits >`
- class `std::__Hashtable<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`

## Enumerations

- enum `{ _S_n_primes }`

## Functions

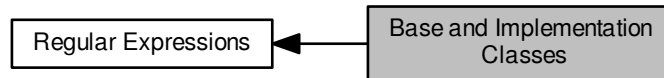
- template<class `_Iterator` >  
`std::iterator_traits<_Iterator>::difference_type std::__detail::__distance_fw (_Iterator __first, _Iterator __last, std::input_iterator_tag)`
- template<class `_Iterator` >  
`std::iterator_traits<_Iterator>::difference_type std::__detail::__distance_fw (_Iterator __first, _Iterator __last, std::forward_iterator_tag)`
- template<class `_Iterator` >  
`std::iterator_traits<_Iterator>::difference_type std::__detail::__distance_fw (_Iterator __first, _Iterator __last)`
- bool `std::__detail::Equality<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >::M_equal (const __hashtable &) const`
- bool `std::__detail::Equality<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >::M_equal (const __hashtable &) const`
- template<typename `_Uiterator` >  
static bool `std::__detail::Equality_base::S_is_permutation (_Uiterator, _Uiterator, _Uiterator)`
- mapped\_type & `std::__detail::Map_base<_Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, __RehashPolicy, _Traits, true >::at (const key_type & __k)`
- const mapped\_type & `std::__detail::Map_base<_Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >::at (const key_type & __k) const`

- `template<typename _InputIterator >`  
`void std::__detail::Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >::insert ( _InputIterator __first, _InputIterator __last)`
- `template<typename _Value, bool _Cache_hash_code>`  
`bool std::__detail::operator!= (const _Node_iterator_base< _Value, _Cache_hash_code > &__x, const _Node_iterator_base< _Value, _Cache_hash_code > &__y)`
- `template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache>`  
`bool std::__detail::operator!= (const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__x, const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__y)`
- `template<typename _Value, bool _Cache_hash_code>`  
`bool std::__detail::operator== (const _Node_iterator_base< _Value, _Cache_hash_code > &__x, const _Node_iterator_base< _Value, _Cache_hash_code > &__y)`
- `template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache>`  
`bool std::__detail::operator== (const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__x, const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__y)`
- `mapped_type & std::__detail::Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >::operator[] (const key_type &__k)`
- `mapped_type & std::__detail::Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >::operator[] (key_type &&__k)`

### 2.8.1 Detailed Description

## 2.9 Base and Implementation Classes

Collaboration diagram for Base and Implementation Classes:



### Classes

- class `std::__detail::_Automaton`
- struct `std::__detail::_CharMatcher< _InIterT, _TraitsT >`
- class `std::__detail::_Compiler< _InIter, _TraitsT >`
- struct `std::__detail::_EndTagger< _FwdIterT, _TraitsT >`
- class `std::__detail::_Grep_matcher`
- class `std::__detail::_Nfa`
- struct `std::__detail::_PatternCursor`
- struct `std::__detail::_RangeMatcher< _InIterT, _TraitsT >`
- struct `std::__detail::_Results`
- class `std::__detail::_Scanner< _InputIterator >`
- struct `std::__detail::_Scanner_base`
- class `std::__detail::_SpecializedCursor< _FwdIterT >`
- class `std::__detail::_SpecializedResults< _FwdIterT, _Alloc >`
- struct `std::__detail::_StartTagger< _FwdIterT, _TraitsT >`
- struct `std::__detail::_State`
- class `std::__detail::_StateSeq`

### Typedefs

- typedef `std::shared_ptr< _Automaton > std::__detail::_AutomatonPtr`
- typedef `std::function< bool(const _PatternCursor &)> std::__detail::_Matcher`
- typedef `int std::__detail::_StateIdT`
- typedef `std::set< _StateIdT > std::__detail::_StateSet`
- typedef `std::stack< _StateIdT, std::vector< _StateIdT > > std::__detail::_StateStack`
- typedef `std::function< void(const _PatternCursor &, _Results &)> std::__detail::_Tagger`

### Enumerations

- enum `std::__detail::_Opcode` {  
`_S_opcode_unknown, _S_opcode_alternative, _S_opcode_subexpr_begin, _S_opcode_subexpr_end,`  
`_S_opcode_match, _S_opcode_accept` }

- enum `std::__detail::Scanner<_InputIterator>::TokenT` {  
`_S_token_anychar`, `_S_token_backref`, `_S_token_bracket_begin`, `_S_token_bracket_end`,  
`_S_token_inverse_class`, `_S_token_char_class_name`, `_S_token_closure0`, `_S_token_closure1`,  
`_S_token_collelem_multi`, `_S_token_collelem_single`, `_S_token_collsymbol`, `_S_token_comma`,  
`_S_token_dash`, `_S_token_dup_count`, `_S_token_eof`, `_S_token_equiv_class_name`,  
`_S_token_interval_begin`, `_S_token_interval_end`, `_S_token_line_begin`, `_S_token_line_end`,  
`_S_token_opt`, `_S_token_or`, `_S_token_ord_char`, `_S_token_quoted_char`,  
`_S_token_subexpr_begin`, `_S_token_subexpr_end`, `_S_token_word_begin`, `_S_token_word_end`,  
`_S_token_unknown` }

## Functions

- `std::__detail::Compiler<_InIter, _TraitsT>::Compiler` (const `_InIter` &\_\_b, const `_InIter` &\_\_e, `_TraitsT` &\_\_traits, `_FlagT` \_\_flags)
- `std::__detail::SpecializedResults<_FwdIterT, _Alloc>::SpecializedResults` (const `_Automaton::SizeT` \_\_size, const `_SpecializedCursor<_FwdIterT>` &\_\_cursor, `match_results<_FwdIterT, _Alloc>` &\_\_m)
- `template<typename _InIter, typename _TraitsT>`  
`_AutomatonPtr std::__detail::compile` (const `_InIter` &\_\_b, const `_InIter` &\_\_e, `_TraitsT` &\_\_t, `regex_constants::syntax_option_type` \_\_f)
- `template<typename _FwdIterT>`  
`_SpecializedCursor<_FwdIterT> std::__detail::cursor` (const `_FwdIterT` &\_\_b, const `_FwdIterT` \_\_e)
- `bool std::__detail::AnyMatcher` (const `_PatternCursor` &)
- `void std::__detail::Scanner<_InputIterator>::M_advance` ()
- `void std::__detail::SpecializedResults<_FwdIterT, _Alloc>::M_set_pos` (int \_\_i, int \_\_j, const `_PatternCursor` &\_\_pc)

## Variables

- `static const _StateIdT std::__detail::S_invalid_state_id`

### 2.9.1 Detailed Description

### 2.9.2 Typedef Documentation

#### 2.9.2.1 `typedef std::shared_ptr<_Automaton> std::__detail::_AutomatonPtr`

Generic shared pointer to an automaton.

Definition at line 62 of file `regex_nfa.h`.

#### 2.9.2.2 `typedef std::function<bool (const _PatternCursor&)> std::__detail::_Matcher`

Indicates if current state matches cursor current.

Definition at line 120 of file `regex_nfa.h`.

#### 2.9.2.3 `typedef int std::__detail::_StateIdT`

Identifies a state in the NFA.

Definition at line 195 of file `regex_nfa.h`.

#### 2.9.2.4 `typedef std::set<_StateIdT> std::__detail::_StateSet`

The Grep Matcher works on sets of states. Here are sets of states.

Definition at line 251 of file `regex_nfa.h`.

#### 2.9.2.5 `typedef std::stack<_StateIdT, std::vector<_StateIdT> > std::__detail::_StateStack`

A stack of states used in evaluating the NFA.

Definition at line 105 of file `regex_grep_matcher.h`.

#### 2.9.2.6 `typedef std::function<void (const _PatternCursor&, _Results&)> std::__detail::_Tagger`

Tags current state (for subexpr begin/end).

Definition at line 84 of file `regex_nfa.h`.

### 2.9.3 Enumeration Type Documentation

#### 2.9.3.1 `enum std::__detail::_Opcode`

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

Definition at line 66 of file `regex_nfa.h`.

#### 2.9.3.2 `template<typename _InputIterator> enum std::__detail::_Scanner::_TokenT`

Token types returned from the scanner.

Definition at line 75 of file `regex_compiler.h`.

### 2.9.4 Function Documentation

#### 2.9.4.1 `bool std::__detail::_AnyMatcher ( const _PatternCursor & ) [inline]`

Matches any character.

Definition at line 124 of file `regex_nfa.h`.

### 2.9.5 Variable Documentation

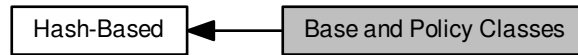
#### 2.9.5.1 `const _StateIdT std::__detail::_S_invalid_state_id [static]`

The special case in which a state identifier is not an index.

Definition at line 198 of file `regex_nfa.h`.

## 2.10 Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



### Classes

- `class __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >`
- `class __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >`

### Enumerations

- `enum { store_hash }`
- `enum { store_hash }`

#### 2.10.1 Detailed Description

#### 2.10.2 Enumeration Type Documentation

2.10.2.1 `template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy> anonymous enum`

Value stores hash, true or false.

Definition at line 200 of file `cc_ht_map.hpp`.

2.10.2.2 `template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy> anonymous enum`

Value stores hash, true or false.

Definition at line 208 of file `gp_ht_map.hpp`.

## 2.11 Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



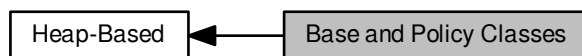
### Classes

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

### 2.11.1 Detailed Description

## 2.12 Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



### Classes

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

### 2.12.1 Detailed Description



## 2.13 Bernoulli Distributions

Collaboration diagram for Bernoulli Distributions:



### Classes

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

### Functions

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

## 2.13.1 Detailed Description

## 2.13.2 Function Documentation

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

Return true if two Bernoulli distributions have different parameters.

Definition at line 3729 of file random.h.

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

Return true if two binomial distributions are different.

Definition at line 3995 of file random.h.

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

Return true if two geometric distributions have different parameters.

Definition at line 4164 of file random.h.

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

Return true if two negative binomial distributions are different.

Definition at line 4409 of file random.h.

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

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

Parameters

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

Returns

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

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

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

Parameters

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

**Returns**

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

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

Extracts a bernoulli\_distribution random number distribution `__x` from the input stream `__is`.

**Parameters**

<code>__is</code>	An input stream.
<code>__x</code>	A bernoulli_distribution random number generator engine.

**Returns**

The input stream with `__x` extracted or in an error state.

Definition at line 3759 of file random.h.

References `std::bernoulli_distribution::param()`.

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

Extracts a geometric\_distribution random number distribution `__x` from the input stream `__is`.

**Parameters**

<code>__is</code>	An input stream.
<code>__x</code>	A geometric_distribution random number generator engine.

**Returns**

The input stream with `__x` extracted or in an error state.

## 2.14 Binary Search

Collaboration diagram for Binary Search:



### Functions

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

### 2.14.1 Detailed Description

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

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

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

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

## 2.14.2 Function Documentation

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

Determines whether an element exists in a range.

### Parameters

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

### Returns

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

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

Definition at line 2697 of file `stl_algo.h`.

References `std::lower_bound()`.

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

Determines whether an element exists in a range.

### Parameters

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

### Returns

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

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

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

Definition at line 2730 of file `stl_algo.h`.

References `std::lower_bound()`.

**2.14.2.3** `template<typename _ForwardIterator, typename _Tp> pair<_ForwardIterator, _ForwardIterator> std::equal_range ( _ForwardIterator __first, _ForwardIterator __last, const _Tp & __val )`

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

### Parameters

\_\_\_\_\_

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

**Returns**

An pair of iterators defining the subrange.

This is equivalent to

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

but does not actually call those functions.

Definition at line 2574 of file `stl_algo.h`.

References `std::advance()`, `std::distance()`, `std::lower_bound()`, and `std::upper_bound()`.

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

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

**Parameters**

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

**Returns**

An pair of iterators defining the subrange.

This is equivalent to

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

but does not actually call those functions.

Definition at line 2636 of file `stl_algo.h`.

References `std::advance()`, `std::distance()`, `std::lower_bound()`, and `std::upper_bound()`.

**2.14.2.5** `template<typename _ForwardIterator, typename _Tp> _ForwardIterator std::lower_bound ( _ForwardIterator __first, _ForwardIterator __last, const _Tp & __val )`

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

**Parameters**

<code>__first</code>	An iterator.
----------------------	--------------

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

**Returns**

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

Definition at line 943 of file `stl_algobase.h`.

References `std::advance()`, and `std::distance()`.

Referenced by `std::__merge_adaptive()`, `std::__merge_without_buffer()`, `std::binary_search()`, and `std::equal_range()`.

**2.14.2.6** `template<typename _ForwardIterator, typename _Tp, typename _Compare> _ForwardIterator std::lower_bound (`  
`_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, _Compare __comp )`

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

**Parameters**

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

**Returns**

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

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

Definition at line 2425 of file `stl_algo.h`.

References `std::advance()`, and `std::distance()`.

**2.14.2.7** `template<typename _ForwardIterator, typename _Tp> _ForwardIterator std::upper_bound (`  
`_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val )`

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

**Parameters**

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

**Returns**

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

Definition at line 2472 of file `stl_algo.h`.

References `std::advance()`, and `std::distance()`.

**2.14.2.8** `template<typename _ForwardIterator, typename _Tp, typename _Compare> _ForwardIterator std::upper_bound (`  
`_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, _Compare __comp )`

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

## Parameters

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

## Returns

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

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

Definition at line 2521 of file `stl_algo.h`.

References `std::advance()`, and `std::distance()`.

Referenced by `std::__merge_adaptive()`, `std::__merge_without_buffer()`, and `std::equal_range()`.



## 2.15 Binder Classes

Collaboration diagram for Binder Classes:



### Classes

- class `std::binder1st<_Operation>`
- class `std::binder2nd<_Operation>`

### Functions

- `template<typename _Operation, typename _Tp>`  
`binder1st<_Operation> std::bind1st(const _Operation &__fn, const _Tp &__x)`
- `template<typename _Operation, typename _Tp>`  
`binder2nd<_Operation> std::bind2nd(const _Operation &__fn, const _Tp &__x)`

#### 2.15.1 Detailed Description

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

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

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

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

These become more useful when combined with the composition functions.

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

#### 2.15.2 Function Documentation

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

One of the [binder functors](#).

Definition at line 131 of file binders.h.

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

One of the [binder functors](#).

Definition at line 166 of file binders.h.

## 2.16 Boolean Operations Classes

Collaboration diagram for Boolean Operations Classes:



### Classes

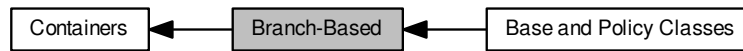
- struct `std::logical_and<_Tp>`
- struct `std::logical_not<_Tp>`
- struct `std::logical_or<_Tp>`

### 2.16.1 Detailed Description

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

## 2.17 Branch-Based

Collaboration diagram for Branch-Based:



### Modules

- [Base and Policy Classes](#)

### Classes

- [class `\_\_gnu\_pbds::basic\_branch`](#)`< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >`
- [class `\_\_gnu\_pbds::tree`](#)`< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >`
- [class `\_\_gnu\_pbds::trie`](#)`< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >`

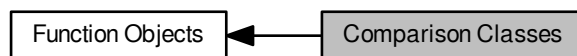
### Macros

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

#### 2.17.1 Detailed Description

## 2.18 Comparison Classes

Collaboration diagram for Comparison Classes:



### Classes

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

### 2.18.1 Detailed Description

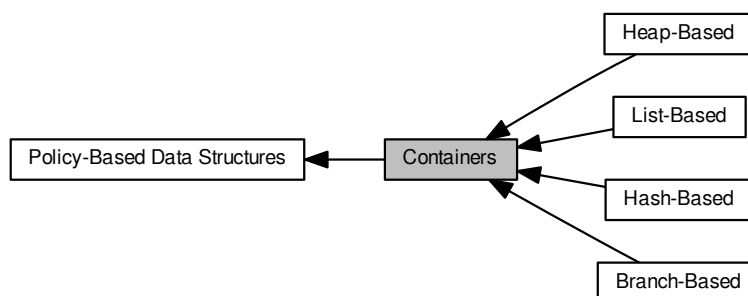
The library provides six wrapper functors for all the basic comparisons in C++, like `<`.

**2.19 Concurrency**

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

## 2.20 Containers

Collaboration diagram for Containers:



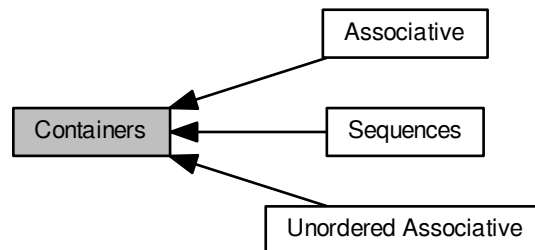
### Modules

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

### 2.20.1 Detailed Description

## 2.21 Containers

Collaboration diagram for Containers:



### Modules

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

#### 2.21.1 Detailed Description

Containers are collections of objects.

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

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

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

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

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



## 2.22 Data Structure Type

Collaboration diagram for Data Structure Type:



### Classes

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

### 2.22.1 Detailed Description

## 2.23 Diagnostics

Components for error handling, reporting, and diagnostic operations.

## 2.24 Exceptions

Collaboration diagram for Exceptions:



### Classes

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

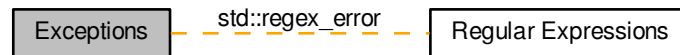
### Functions

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

### 2.24.1 Detailed Description

## 2.25 Exceptions

Collaboration diagram for Exceptions:



### Classes

- class [\\_\\_cxxabiv1::\\_\\_forced\\_unwind](#)
- struct [\\_\\_gnu\\_cxx::forced\\_error](#)
- class [\\_\\_gnu\\_cxx::recursive\\_init\\_error](#)
- class [std::\\_\\_exception\\_ptr::exception\\_ptr](#)
- class [std::bad\\_weak\\_ptr](#)
- class [std::ios\\_base::failure](#)
- class [std::nested\\_exception](#)
- class [std::regex\\_error](#)

### Functions

- template<typename \_Ex >  
const nested\_exception \* [std::\\_\\_get\\_nested\\_exception](#) (const \_Ex &\_\_ex)
- template<typename \_Ex >  
void [std::\\_\\_throw\\_with\\_nested](#) (\_Ex &&, const nested\_exception \*=0) [\\_\\_attribute\\_\\_\(\(\\_\\_noreturn\\_\\_\)\)](#)
- template<typename \_Ex >  
void [std::\\_\\_throw\\_with\\_nested](#) (\_Ex &&,...) [\\_\\_attribute\\_\\_\(\(\\_\\_noreturn\\_\\_\)\)](#)
- template<typename \_Ex >  
exception\_ptr [std::copy\\_exception](#) (\_Ex \_\_ex) noexcept
- exception\_ptr [std::current\\_exception](#) () noexcept
- template<typename \_Ex >  
exception\_ptr [std::make\\_exception\\_ptr](#) (\_Ex \_\_ex) noexcept
- void [std::rethrow\\_exception](#) (exception\_ptr) [\\_\\_attribute\\_\\_\(\(\\_\\_noreturn\\_\\_\)\)](#)
- template<typename \_Ex >  
void [std::rethrow\\_if\\_nested](#) (const \_Ex &\_\_ex)
- void [std::rethrow\\_if\\_nested](#) (const nested\_exception &\_\_ex)
- template<typename \_Ex >  
void [std::throw\\_with\\_nested](#) (\_Ex \_\_ex)

#### 2.25.1 Detailed Description

#### 2.25.2 Function Documentation

**2.25.2.1** `template<typename _Ex > exception_ptr std::copy_exception ( _Ex __ex ) [noexcept]`

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

Definition at line 169 of file `exception_ptr.h`.

References `std::current_exception()`.

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

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

Referenced by `std::copy_exception()`.

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

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

Definition at line 188 of file `exception_ptr.h`.

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

Throw the object pointed to by the `exception_ptr`.

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

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

Definition at line 146 of file `nested_exception.h`.

**2.25.2.6** `void std::rethrow_if_nested ( const nested_exception & __ex ) [inline]`

Overload, See N2619.

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

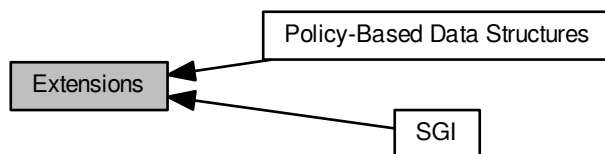
**2.25.2.7** `template<typename _Ex > void std::throw_with_nested ( _Ex __ex ) [inline]`

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

Definition at line 136 of file `nested_exception.h`.

## 2.26 Extensions

Collaboration diagram for Extensions:



### Modules

- [Policy-Based Data Structures](#)
- [SGI](#)

### Classes

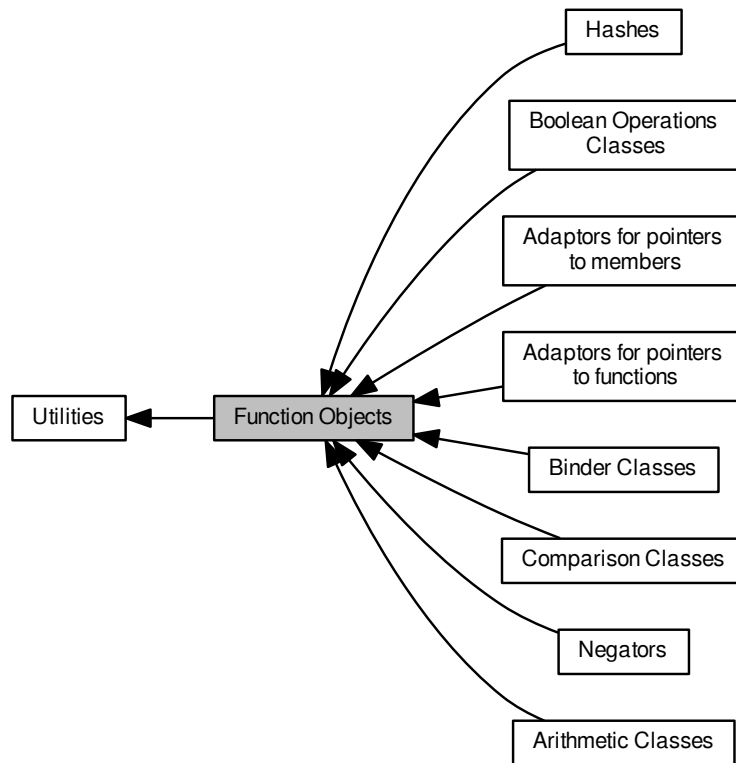
- [class `\_\_gnu\_cxx::\_\_versa\_string<\_CharT, \_Traits, \_Alloc, \_Base>`](#)

#### 2.26.1 Detailed Description

Components generally useful that are not part of any standard.

## 2.27 Function Objects

Collaboration diagram for Function Objects:



### Modules

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

### Classes

- `struct std::binary_function< _Arg1, _Arg2, _Result >`
- `struct std::unary_function< _Arg, _Result >`

### 2.27.1 Detailed Description

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

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

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

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

To negate every element in `a`, use

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

The addition and negation functions will be inlined directly.

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



## 2.28 Hash-Based

Collaboration diagram for Hash-Based:



### Modules

- [Base and Policy Classes](#)

### Classes

- `class __gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Ti, _Alloc >`
- `class __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >`
- `class __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >`

### Macros

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

#### 2.28.1 Detailed Description

## 2.29 Hashes

Collaboration diagram for Hashes:



### Classes

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

### Macros

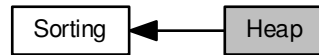
- `#define _Cxx_hashtable_define_trivial_hash(_Tp)`

#### 2.29.1 Detailed Description

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

## 2.30 Heap

Collaboration diagram for Heap:



### Functions

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

#### 2.30.1 Detailed Description

#### 2.30.2 Function Documentation

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

Determines whether a range is a heap.

## Parameters

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

## Returns

True if range is a heap, false otherwise.

Definition at line 571 of file `stl_heap.h`.

References `std::is_heap_until()`.

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

Determines whether a range is a heap using comparison functor.

## Parameters

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

## Returns

True if range is a heap, false otherwise.

Definition at line 584 of file `stl_heap.h`.

References `std::is_heap_until()`.

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

Search the end of a heap.

## Parameters

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

## Returns

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

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

Definition at line 523 of file `stl_heap.h`.

References `std::distance()`.

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

Search the end of a heap using comparison functor.

**Parameters**

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

**Returns**

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

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

Definition at line 549 of file `stl_heap.h`.

References `std::distance()`.

Referenced by `std::is_heap()`.

**2.30.2.5** `template<typename _RandomAccessIterator > void std::make_heap ( _RandomAccessIterator __first, _RandomAccessIterator __last )`

Construct a heap over a range.

**Parameters**

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

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

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

**2.30.2.6** `template<typename _RandomAccessIterator, typename _Compare > void std::make_heap ( _RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp )`

Construct a heap over a range using comparison functor.

**Parameters**

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

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

Definition at line 426 of file `stl_heap.h`.

Referenced by `std::__heap_select()`, `std::partial_sort_copy()`, and `std::priority_queue< _Tp, _Sequence, _Compare >::priority_queue()`.

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

Pop an element off a heap.

**Parameters**


---

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

**Precondition**

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

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

Definition at line 281 of file `stl_heap.h`.

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

Pop an element off a heap using comparison functor.

**Parameters**

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

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

Definition at line 359 of file `stl_heap.h`.

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

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

Push an element onto a heap.

**Parameters**

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

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

Definition at line 156 of file `stl_heap.h`.

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

Push an element onto a heap using comparison functor.

**Parameters**

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

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

Definition at line 206 of file `stl_heap.h`.

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

2.30.2.11 `template<typename _RandomAccessIterator > void std::sort_heap ( _RandomAccessIterator __first,  
_RandomAccessIterator __last )`

Sort a heap.

## Parameters

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

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

Definition at line 465 of file `stl_heap.h`.

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

Sort a heap using comparison functor.

## Parameters

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

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

Definition at line 494 of file `stl_heap.h`.

Referenced by `std::partial_sort()`, and `std::partial_sort_copy()`.



## 2.31 Heap-Based

Collaboration diagram for Heap-Based:



### Modules

- [Base and Policy Classes](#)

### Classes

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

### Typedefs

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

### Functions

- [\\_\\_gnu\\_pbds::priority\\_queue<\\_Tv, Cmp\\_Fn, Tag, \\_Alloc>::priority\\_queue](#) (const cmp\_fn &r\_cmp\_fn)
- template<typename It >  
[\\_\\_gnu\\_pbds::priority\\_queue<\\_Tv, Cmp\\_Fn, Tag, \\_Alloc>::priority\\_queue](#) (It first\_it, It last\_it)
- template<typename It >  
[\\_\\_gnu\\_pbds::priority\\_queue<\\_Tv, Cmp\\_Fn, Tag, \\_Alloc>::priority\\_queue](#) (It first\_it, It last\_it, const cmp\_fn &r\_cmp\_fn)

- `__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::priority_queue` (const priority\_queue &other)
- `priority_queue & __gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::operator=` (const priority\_queue &other)
- `void __gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::swap` (priority\_queue &other)

### 2.31.1 Detailed Description

### 2.31.2 Function Documentation

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

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

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

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

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

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

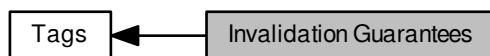
2.31.2.3 `template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::priority_queue ( It first_it, It last_it, const cmp_fn & r_cmp_fn ) [inline]`

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

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

## 2.32 Invalidation Guarantees

Collaboration diagram for Invalidation Guarantees:



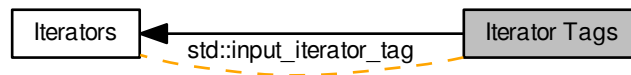
### Classes

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

### 2.32.1 Detailed Description

## 2.33 Iterator Tags

Collaboration diagram for Iterator Tags:



### Classes

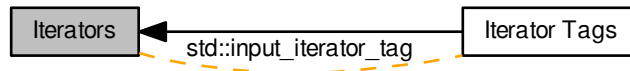
- struct [std::bidirectional\\_iterator\\_tag](#)
- struct [std::forward\\_iterator\\_tag](#)
- struct [std::input\\_iterator\\_tag](#)
- struct [std::output\\_iterator\\_tag](#)
- struct [std::random\\_access\\_iterator\\_tag](#)

#### 2.33.1 Detailed Description

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

## 2.34 Iterators

Collaboration diagram for Iterators:



### Modules

- [Iterator Tags](#)

### Classes

- class [std::\\_\\_has\\_iterator\\_category\\_helper< \\_Tp >](#)
- class [std::back\\_insert\\_iterator< \\_Container >](#)
- class [std::front\\_insert\\_iterator< \\_Container >](#)
- struct [std::input\\_iterator\\_tag](#)
- class [std::insert\\_iterator< \\_Container >](#)
- class [std::istream\\_iterator< \\_Tp, \\_CharT, \\_Traits, \\_Dist >](#)
- class [std::istreambuf\\_iterator< \\_CharT, \\_Traits >](#)
- struct [std::iterator< \\_Category, \\_Tp, \\_Distance, \\_Pointer, \\_Reference >](#)
- struct [std::iterator\\_traits< \\_Tp \\* >](#)
- struct [std::iterator\\_traits< const \\_Tp \\* >](#)
- class [std::move\\_iterator< \\_Iterator >](#)
- class [std::ostream\\_iterator< \\_Tp, \\_CharT, \\_Traits >](#)
- class [std::ostreambuf\\_iterator< \\_CharT, \\_Traits >](#)
- class [std::reverse\\_iterator< \\_Iterator >](#)

### Functions

- [template<bool \\_IsMove, typename \\_CharT >](#)  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if< \\_\\_is\\_char< \\_CharT >::\\_\\_value, ostreambuf\\_iterator< \\_CharT > >::\\_\\_type std::\\_\\_](#)  
**copy\_move\_a2** ( \_CharT \* \_\_first, \_CharT \* \_\_last, ostreambuf\_iterator< \_CharT > \_\_result)
- [template<bool \\_IsMove, typename \\_CharT >](#)  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if< \\_\\_is\\_char< \\_CharT >::\\_\\_value, ostreambuf\\_iterator< \\_CharT > >::\\_\\_type std::\\_\\_](#)  
**copy\_move\_a2** (const \_CharT \* \_\_first, const \_CharT \* \_\_last, ostreambuf\_iterator< \_CharT > \_\_result)
- [template<bool \\_IsMove, typename \\_CharT >](#)  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if< \\_\\_is\\_char< \\_CharT >::\\_\\_value, \\_CharT \\* >::\\_\\_type std::\\_\\_](#)  
**copy\_move\_a2** (istreambuf\_iterator< \_CharT > \_\_first, istreambuf\_iterator< \_CharT > \_\_last, \_CharT \* \_\_result)
- [template<typename \\_Iter >](#)  
[iterator\\_traits< \\_Iter >::iterator\\_category](#) [std::iterator\\_category](#) (const \_Iter &)
- [template<typename \\_Iterator, typename \\_ReturnType = typename conditional< \\_\\_move\\_if\\_noexcept\\_cond <typename iterator\\_traits< \\_](#)  
[Iterator>::value\\_type>::value, \\_Iterator, move\\_iterator< \\_Iterator > >::type>](#)  
**\_ReturnType std::\_\_make\_move\_if\_noexcept\_iterator** ( \_Iterator \_\_i)

- `template<typename _Container >`  
`back_insert_iterator< _Container > std::back\_inserter ( _Container &__x)`
- `template<typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::copy`  
`(istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT >`  
`__result)`
- `template<typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf_iterator< _CharT > >::__type std::find`  
`(istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT &__val)`
- `template<typename _Container >`  
`front_insert_iterator< _Container > std::front\_inserter ( _Container &__x)`
- `template<typename _Container , typename _Iterator >`  
`insert_iterator< _Container > std::inserter ( _Container &__x, _Iterator __i)`
- `template<typename _Iterator >`  
`move_iterator< _Iterator > std::make_move_iterator ( _Iterator __i)`
- `template<class _Tp , class _CharT , class _Traits , class _Dist >`  
`bool std::operator!= (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp,`  
`_CharT, _Traits, _Dist > &__y)`
- `template<typename _CharT , typename _Traits >`  
`bool std::operator!= (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT,`  
`_Traits > &__b)`
- `template<typename _Iterator >`  
`bool std::operator!= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL , typename _IteratorR >`  
`bool std::operator!= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL , typename _IteratorR >`  
`bool std::operator!= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool std::operator!= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`  
`reverse_iterator< _Iterator > std::operator+ (typename reverse_iterator< _Iterator >::difference_type __n,`  
`const reverse_iterator< _Iterator > &__x)`
- `template<typename _Iterator >`  
`move_iterator< _Iterator > std::operator+ (typename move_iterator< _Iterator >::difference_type __n, const`  
`move_iterator< _Iterator > &__x)`
- `template<typename _Iterator >`  
`reverse_iterator< _Iterator >::difference_type std::operator- (const reverse_iterator< _Iterator > &__x, const`  
`reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL , typename _IteratorR >`  
`auto std::operator- (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y) ->`  
`decltype(__y.base()-__x.base())`
- `template<typename _IteratorL , typename _IteratorR >`  
`auto std::operator- (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y) ->`  
`decltype(__x.base()-__y.base())`
- `template<typename _Iterator >`  
`auto std::operator- (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y) ->`  
`decltype(__x.base()-__y.base())`
- `template<typename _Iterator >`  
`bool std::operator< (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL , typename _IteratorR >`  
`bool std::operator< (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL , typename _IteratorR >`  
`bool std::operator< (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`

- `template<typename _Iterator >`  
`bool std::operator< (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`  
`bool std::operator<= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator<= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator<= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool std::operator<= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Tp, typename _CharT, typename _Traits, typename _Dist >`  
`bool std::operator== (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`
- `template<typename _CharT, typename _Traits >`  
`bool std::operator== (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)`
- `template<typename _Iterator >`  
`bool std::operator== (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator== (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator== (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool std::operator== (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`  
`bool std::operator> (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator> (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator> (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool std::operator> (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`  
`bool std::operator>= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator>= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator>= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool std::operator>= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`

### 2.34.1 Detailed Description

Abstractions for uniform iterating through various underlying types.

### 2.34.2 Function Documentation

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

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

Definition at line 201 of file `stl_iterator_base_types.h`.

Referenced by `std::__find_if_not()`, `__gnu_debug::__valid_range_aux()`, `std::advance()`, `std::copy_n()`, `std::distance()`, `std::find()`, `std::find_end()`, `std::find_if()`, `std::partition()`, `std::reverse()`, `std::search_n()`, `std::uninitialized_copy_n()`, and `std::unique_copy()`.

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

#### Parameters

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

#### Returns

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

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

Definition at line 479 of file `stl_iterator.h`.

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

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

#### Parameters

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

#### Returns

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

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

Definition at line 569 of file `stl_iterator.h`.

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

#### Parameters

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

#### Returns

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

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

Definition at line 683 of file `stl_iterator.h`.



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

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

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

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

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

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

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

#### Parameters

<code>__x</code>	A reverse_iterator.
<code>__y</code>	A reverse_iterator.

#### Returns

A simple bool.

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

Definition at line 291 of file stl\_iterator.h.

References `std::reverse_iterator< _Iterator >::base()`.

## 2.35 List-Based

Collaboration diagram for List-Based:



### Classes

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

### Macros

- `#define PB_DS_LU_BASE`

### 2.35.1 Detailed Description

## 2.36 Locales

### Classes

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

### Enumerations

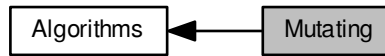
- enum `{ _S_minus, _S_zero, _S_end }`
- enum `dateorder { no_order, dmy, mdy, ymd, ydm }`
- enum `part { none, space, symbol, sign, value }`

#### 2.36.1 Detailed Description

Classes and functions for internationalization and localization.

## 2.37 Mutating

Collaboration diagram for Mutating:



## Functions

- `template<typename _II, typename _OI >`  
`_OI std::copy (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`  
`_BI2 std::copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`_OutputIterator std::copy_n (_InputIterator __first, _Size __n, _OutputIterator __result)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void std::fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _OI, typename _Size, typename _Tp >`  
`_OI std::fill_n (_OI __first, _Size __n, const _Tp &__value)`
- `template<typename _ForwardIterator, typename _Generator >`  
`void std::generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator std::generate_n (_OutputIterator __first, _Size __n, _Generator __gen)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool std::is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`void std::iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _II, typename _OI >`  
`_OI std::move (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`  
`_BI2 std::move_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate >`  
`pair< _OutputIterator1, _OutputIterator2 > std::partition_copy (_InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::partition_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _RandomAccessIterator >`  
`void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator >`  
`void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator &&__rand)`

- `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator std::remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`  
`_OutputIterator std::remove\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::remove\_copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result,  $\leftrightarrow$  Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::remove\_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void std::replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__old_value, const _Tp &__new $\leftrightarrow$  _value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp >`  
`_OutputIterator std::replace\_copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result,  $\leftrightarrow$  Predicate __pred, const _Tp &__new_value)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Tp >`  
`void std::replace\_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp &__new $\leftrightarrow$  value)`
- `template<typename _BidirectionalIterator >`  
`void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _OutputIterator >`  
`_OutputIterator std::reverse\_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator  $\leftrightarrow$  __result)`
- `template<typename _ForwardIterator >`  
`void std::rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _OutputIterator >`  
`_OutputIterator std::rotate\_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last,  $\leftrightarrow$  _OutputIterator __result)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator >`  
`void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumber $\leftrightarrow$  Generator &&__g)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::stable\_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator2 std::swap\_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2  $\leftrightarrow$  __first2)`
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Unary $\leftrightarrow$  Operation __unary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Output $\leftrightarrow$  Iterator __result, _BinaryOperation __binary_op)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _OutputIterator >`  
`_OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`  
`_OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Binary $\leftrightarrow$  Predicate __binary_pred)`

## 2.37.1 Detailed Description

## 2.37.2 Function Documentation

2.37.2.1 `template<typename _II, typename _OI> _OI std::copy ( _II __first, _II __last, _OI __result ) [inline]`

Copies the range [first,last) into result.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.

## Returns

`result + (first - last)`

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling). Result may not be contained within [first,last); the `copy_backward` function should be used instead.

Note that the end of the output range is permitted to be contained within [first,last).

Definition at line 450 of file `stl_algobase.h`.

2.37.2.2 `template<typename _BI1, typename _BI2> _BI2 std::copy_backward ( _BI1 __first, _BI1 __last, _BI2 __result ) [inline]`

Copies the range [first,last) into result.

## Parameters

<code>__first</code>	A bidirectional iterator.
<code>__last</code>	A bidirectional iterator.
<code>__result</code>	A bidirectional iterator.

## Returns

`result - (first - last)`

The function has the same effect as `copy`, but starts at the end of the range and works its way to the start, returning the start of the result. This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Result may not be in the range (first,last]. Use `copy` instead. Note that the start of the output range may overlap [first,last).

Definition at line 619 of file `stl_algobase.h`.

2.37.2.3 `template<typename _InputIterator, typename _OutputIterator, typename _Predicate> _OutputIterator std::copy_if ( _InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred )`

Copy the elements of a sequence for which a predicate is true.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__pred</code>	A predicate.

**Returns**

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first,__last)` for which `__pred` returns true to the range beginning at `__result`. `copy_if()` is stable, so the relative order of elements that are copied is unchanged.

Definition at line 950 of file `stl_algo.h`.

```
2.37.2.4 template<typename _InputIterator, typename _Size, typename _OutputIterator > _OutputIterator std::copy_n (
    _InputIterator __first, _Size __n, _OutputIterator __result ) [inline]
```

Copies the range `[first,first+n)` into `[result,result+n)`.

**Parameters**

<code>__first</code>	An input iterator.
<code>__n</code>	The number of elements to copy.
<code>__result</code>	An output iterator.

**Returns**

`result+n`.

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Definition at line 1013 of file `stl_algo.h`.

References `std::__iterator_category()`.

```
2.37.2.5 template<typename _ForwardIterator, typename _Tp > void std::fill ( _ForwardIterator __first, _ForwardIterator __last,
    const _Tp & __value ) [inline]
```

Fills the range `[first,last)` with copies of `value`.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__value</code>	A reference-to-const of arbitrary type.

**Returns**

Nothing.

This function fills a range with copies of the same value. For char types filling contiguous areas of memory, this becomes an inline call to `memset` or `wmemset`.

Definition at line 721 of file `stl_algobase.h`.

```
2.37.2.6  template<typename _OI, typename _Size, typename _Tp > _OI std::fill_n ( _OI __first, _Size __n, const _Tp & __value )  
          [inline]
```

Fills the range [first,first+n) with copies of value.



**Parameters**

<code>__first</code>	An output iterator.
<code>__n</code>	The count of copies to perform.
<code>__value</code>	A reference-to-const of arbitrary type.

**Returns**

The iterator at `first+n`.

This function fills a range with copies of the same value. For char types filling contiguous areas of memory, this becomes an inline call to `memset` or `@ wmemset`.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 865. More algorithms that throw away information

Definition at line 781 of file `stl_algobase.h`.

**2.37.2.7** `template<typename _ForwardIterator, typename _Generator> void std::generate ( _ForwardIterator __first, _ForwardIterator __last, _Generator __gen )`

Assign the result of a function object to each value in a sequence.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__gen</code>	A function object taking no arguments and returning <code>std::iterator_traits&lt;_ForwardIterator&gt;::value_type</code>

**Returns**

`generate()` returns no value.

Performs the assignment `*i = __gen()` for each `i` in the range `[__first, __last)`.

Definition at line 5048 of file `stl_algo.h`.

**2.37.2.8** `template<typename _OutputIterator, typename _Size, typename _Generator> _OutputIterator std::generate_n ( _OutputIterator __first, _Size __n, _Generator __gen )`

Assign the result of a function object to each value in a sequence.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__n</code>	The length of the sequence.
<code>__gen</code>	A function object taking no arguments and returning <code>std::iterator_traits&lt;_ForwardIterator&gt;::value_type</code>

**Returns**

The end of the sequence, `__first+__n`

Performs the assignment `*i = __gen()` for each `i` in the range `[__first, __first+__n)`.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 865. More algorithms that throw away information

Definition at line 5079 of file `stl_algo.h`.

```
2.37.2.9  template<typename _InputIterator, typename _Predicate > bool std::is_partitioned ( _InputIterator __first, _InputIterator  
        __last, _Predicate __pred ) [inline]
```

Checks whether the sequence is partitioned.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

## Returns

True if the range `[__first,__last)` is partitioned by `__pred`, i.e. if all elements that satisfy `__pred` appear before those that do not.

Definition at line 802 of file `stl_algo.h`.

References `std::find_if_not()`, and `std::none_of()`.

**2.37.2.10** `template<typename _ForwardIterator1, typename _ForwardIterator2 > void std::iter_swap ( _ForwardIterator1 __a, _ForwardIterator2 __b ) [inline]`

Swaps the contents of two iterators.

## Parameters

<code>__a</code>	An iterator.
<code>__b</code>	Another iterator.

## Returns

Nothing.

This function swaps the values pointed to by two iterators, not the iterators themselves.

Definition at line 119 of file `stl_algobase.h`.

References `std::swap()`.

Referenced by `std::__merge_without_buffer()`, `std::__move_median_to_first()`, `std::__partition()`, `std::__reverse()`, `std::__rotate()`, `std::__unguarded_partition()`, `std::next_permutation()`, `std::prev_permutation()`, `std::random_shuffle()`, `std::shuffle()`, and `std::swap_ranges()`.

**2.37.2.11** `template<typename _II, typename _OI > _OI std::move ( _II __first, _II __last, _OI __result ) [inline]`

Moves the range `[first,last)` into `result`.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.

## Returns

`result + (first - last)`

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling). Result may not be contained within `[first,last)`; the `move_backward` function should be used instead.

Note that the end of the output range is permitted to be contained within `[first,last)`.

Definition at line 483 of file `stl_algobase.h`.

```
2.37.2.12  template<typename _BI1, typename _BI2 > _BI2 std::move_backward ( _BI1 __first, _BI1 __last, _BI2 __result )  
           [inline]
```

Moves the range [first,last) into result.

**Parameters**

<code>__first</code>	A bidirectional iterator.
<code>__last</code>	A bidirectional iterator.
<code>__result</code>	A bidirectional iterator.

**Returns**

result - (first - last)

The function has the same effect as `move`, but starts at the end of the range and works its way to the start, returning the start of the result. This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Result may not be in the range `(first,last]`. Use `move` instead. Note that the start of the output range may overlap `[first,last)`.

Definition at line 655 of file `stl_algobase.h`.

**2.37.2.13** `template<typename _ForwardIterator, typename _Predicate> _ForwardIterator std::partition ( _ForwardIterator __first, _ForwardIterator __last, _Predicate __pred ) [inline]`

Move elements for which a predicate is true to the beginning of a sequence.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__pred</code>	A predicate functor.

**Returns**

An iterator `middle` such that `__pred(i)` is true for each iterator `i` in the range `[__first,middle)` and false for each `i` in the range `[middle,__last)`.

`__pred` must not modify its operand. `partition()` does not preserve the relative ordering of elements in each group, use `stable_partition()` if this is needed.

Definition at line 5260 of file `stl_algo.h`.

References `std::iterator_category()`, and `std::__partition()`.

**2.37.2.14** `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate> pair<_OutputIterator1, _OutputIterator2> std::partition_copy ( _InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred )`

Copy the elements of a sequence to separate output sequences depending on the truth value of a predicate.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__out_true</code>	An output iterator.

<code>__out_false</code>	An output iterator.
<code>__pred</code>	A predicate.

**Returns**

A pair designating the ends of the resulting sequences.

Copies each element in the range `[__first,__last)` for which `__pred` returns true to the range beginning at `out_true` and each element for which `__pred` returns false to `__out_false`.

Definition at line 1042 of file `stl_algo.h`.

**2.37.2.15** `template<typename _ForwardIterator, typename _Predicate> _ForwardIterator std::partition_point ( _ForwardIterator __first, _ForwardIterator __last, _Predicate __pred )`

Find the partition point of a partitioned range.

**Parameters**

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__pred</code>	A predicate.

**Returns**

An iterator `mid` such that `all_of(__first, mid, __pred)` and `none_of(mid, __last, __pred)` are both true.

Definition at line 820 of file `stl_algo.h`.

References `std::advance()`, and `std::distance()`.

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

Randomly shuffle the elements of a sequence.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.

**Returns**

Nothing.

Reorder the elements in the range `[__first,__last)` using a random distribution, so that every possible ordering of the sequence is equally likely.

Definition at line 5187 of file `stl_algo.h`.

References `std::iter_swap()`.

**2.37.2.17** `template<typename _RandomAccessIterator, typename _RandomNumberGenerator> void std::random_shuffle ( _RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator && __rand )`

Shuffle the elements of a sequence using a random number generator.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__rand</code>	The RNG functor or function.

**Returns**

Nothing.

Reorders the elements in the range `[__first,__last)` using `__rand` to provide a random distribution. Calling `__rand(↵N)` for a positive integer `N` should return a randomly chosen integer from the range `[0,N)`.

Definition at line 5220 of file `stl_algo.h`.

References `std::iter_swap()`.

**2.37.2.18** `template<typename _ForwardIterator, typename _Tp> _ForwardIterator std::remove ( _ForwardIterator __first, _ForwardIterator __last, const _Tp & __value )`

Remove elements from a sequence.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__value</code>	The value to be removed.

**Returns**

An iterator designating the end of the resulting sequence.

All elements equal to `__value` are removed from the range `[__first,__last)`.

`remove()` is stable, so the relative order of elements that are not removed is unchanged.

Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

Definition at line 1091 of file `stl_algo.h`.

References `std::find()`.

**2.37.2.19** `template<typename _InputIterator, typename _OutputIterator, typename _Tp> _OutputIterator std::remove_copy ( _InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp & __value )`

Copy a sequence, removing elements of a given value.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__value</code>	The value to be removed.

**Returns**

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first,__last)` not equal to `__value` to the range beginning at `__result`. `remove_copy()` is stable, so the relative order of elements that are copied is unchanged.

Definition at line 873 of file `stl_algo.h`.

2.37.2.20 `template<typename _InputIterator , typename _OutputIterator , typename _Predicate > _OutputIterator  
std::remove_copy_if ( _InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred )`

Copy a sequence, removing elements for which a predicate is true.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__pred</code>	A predicate.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first,__last)` for which `__pred` returns false to the range beginning at `__result`.  
`remove_copy_if()` is stable, so the relative order of elements that are copied is unchanged.

Definition at line 911 of file `stl_algo.h`.

2.37.2.21 `template<typename _ForwardIterator , typename _Predicate > _ForwardIterator std::remove_if ( _ForwardIterator __first,  
_ForwardIterator __last, _Predicate __pred )`

Remove elements from a sequence using a predicate.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__pred</code>	A predicate.

Returns

An iterator designating the end of the resulting sequence.

All elements for which `__pred` returns true are removed from the range `[__first,__last)`.

`remove_if()` is stable, so the relative order of elements that are not removed is unchanged.

Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

Definition at line 1134 of file `stl_algo.h`.

References `std::find_if()`.

2.37.2.22 `template<typename _ForwardIterator , typename _Tp > void std::replace ( _ForwardIterator __first, _ForwardIterator  
__last, const _Tp & __old_value, const _Tp & __new_value )`

Replace each occurrence of one value in a sequence with another value.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.



<code>__old_value</code>	The value to be replaced.
<code>__new_value</code>	The replacement value.

**Returns**

`replace()` returns no value.

For each iterator `i` in the range `[__first,__last)` if `*i == __old_value` then the assignment `*i = __new_value` is performed.

Definition at line 4984 of file `stl_algo.h`.

**2.37.2.23** `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp > _OutputIterator  
std::replace_copy_if ( _InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp  
& __new_value )`

Copy a sequence, replacing each value for which a predicate returns true with another value.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__pred</code>	A predicate.
<code>__new_value</code>	The replacement value.

**Returns**

The end of the output sequence, `__result+(__last-__first)`.

Copies each element in the range `[__first,__last)` to the range `[__result,__result+(__last-__first))` replacing elements for which `__pred` returns true with `__new_value`.

Definition at line 3922 of file `stl_algo.h`.

**2.37.2.24** `template<typename _ForwardIterator, typename _Predicate, typename _Tp > void std::replace_if ( _ForwardIterator  
__first, _ForwardIterator __last, _Predicate __pred, const _Tp & __new_value )`

Replace each value in a sequence for which a predicate returns true with another value.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__pred</code>	A predicate.
<code>__new_value</code>	The replacement value.

**Returns**

`replace_if()` returns no value.

For each iterator `i` in the range `[__first,__last)` if `__pred(*i)` is true then the assignment `*i = __new_value` is performed.

Definition at line 5016 of file `stl_algo.h`.

```
2.37.2.25  template<typename _BidirectionalIterator > void std::reverse ( _BidirectionalIterator __first, _BidirectionalIterator __last  
    ) [inline]
```

Reverse a sequence.

**Parameters**

<code>__first</code>	A bidirectional iterator.
<code>__last</code>	A bidirectional iterator.

**Returns**

`reverse()` returns no value.

Reverses the order of the elements in the range `[__first, __last)`, so that the first element becomes the last etc. For every `i` such that  $0 \leq i < (\text{__last} - \text{__first})/2$ , `reverse()` swaps `*(__first+i)` and `*(__last-(i+1))`

Definition at line 1442 of file `stl_algo.h`.

References `std::iterator_category()`, and `std::__reverse()`.

Referenced by `std::next_permutation()`, and `std::prev_permutation()`.

**2.37.2.26** `template<typename _BidirectionalIterator, typename _OutputIterator> _OutputIterator std::reverse_copy (`  
`_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result )`

Copy a sequence, reversing its elements.

**Parameters**

<code>__first</code>	A bidirectional iterator.
<code>__last</code>	A bidirectional iterator.
<code>__result</code>	An output iterator.

**Returns**

An iterator designating the end of the resulting sequence.

Copies the elements in the range `[__first, __last)` to the range `[__result, __result+(__last-__first))` such that the order of the elements is reversed. For every `i` such that  $0 \leq i < (\text{__last} - \text{__first})$ , `reverse_copy()` performs the assignment `*(__result+(__last-__first)-1-i) = *(__first+i)`. The ranges `[__first, __last)` and `[__result, __result+(__last-__first))` must not overlap.

Definition at line 1469 of file `stl_algo.h`.

**2.37.2.27** `template<typename _ForwardIterator> void std::rotate ( _ForwardIterator __first, _ForwardIterator __middle,`  
`_FowardIterator __last ) [inline]`

Rotate the elements of a sequence.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__middle</code>	A forward iterator.
<code>__last</code>	A forward iterator.

**Returns**

Nothing.

Rotates the elements of the range `[__first, __last)` by `(__middle - __first)` positions so that the element at `__middle` is moved to `__first`, the element at `__middle+1` is moved to `__first+1` and so on for each element in the range `[__first, __last)`.

This effectively swaps the ranges [`__first`,`__middle`) and [`__middle`,`__last`).

Performs  $*(\_\text{first} + (n + (\_\text{last} - \_\text{middle})) \% (\_\text{last} - \_\text{first})) = *(\_\text{first} + n)$  for each `n` in the range `[0, __last - __first)`.

Definition at line 1675 of file `stl_algo.h`.

References `std::__rotate()`.

Referenced by `std::__inplace_stable_partition()`, `std::__merge_without_buffer()`, `std::__rotate_adaptive()`, and `std::__↔ stable_partition_adaptive()`.

**2.37.2.28** `template<typename _ForwardIterator, typename _OutputIterator> _OutputIterator std::rotate_copy ( _ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, _OutputIterator __result )`

Copy a sequence, rotating its elements.

#### Parameters

<code>__first</code>	A forward iterator.
<code>__middle</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__result</code>	An output iterator.

#### Returns

An iterator designating the end of the resulting sequence.

Copies the elements of the range [`__first`,`__last`) to the range beginning at

#### Returns

, rotating the copied elements by  $(\_\text{middle} - \_\text{first})$  positions so that the element at `__middle` is moved to `↔ __result`, the element at `__middle+1` is moved to `__result+1` and so on for each element in the range [`__first`,`__last`).

Performs  $*(\_\text{result} + (n + (\_\text{last} - \_\text{middle})) \% (\_\text{last} - \_\text{first})) = *(\_\text{first} + n)$  for each `n` in the range `[0, __last - __first)`.

Definition at line 1711 of file `stl_algo.h`.

**2.37.2.29** `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator> void std::shuffle ( _RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumberGenerator && __g )`

Shuffle the elements of a sequence using a uniform random number generator.

#### Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__g</code>	A <code>UniformRandomNumberGenerator</code> (26.5.1.3).

#### Returns

Nothing.

Reorders the elements in the range [`__first`,`__last`) using `__g` to provide random numbers.

Definition at line 4367 of file `stl_algo.h`.

References `std::iter_swap()`.

2.37.2.30 `template<typename _ForwardIterator, typename _Predicate> _ForwardIterator std::stable_partition ( _ForwardIterator __first, _ForwardIterator __last, _Predicate __pred )`

Move elements for which a predicate is true to the beginning of a sequence, preserving relative ordering.

## Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__pred</code>	A predicate functor.

## Returns

An iterator `middle` such that `__pred(i)` is true for each iterator `i` in the range `[first,middle)` and false for each `i` in the range `[middle,last)`.

Performs the same function as `partition()` with the additional guarantee that the relative ordering of elements in each group is preserved, so any two elements `x` and `y` in the range `[__first,__last)` such that `__pred(x) == __pred(y)` will have the same relative ordering after calling `stable_partition()`.

Definition at line 1890 of file `stl_algo.h`.

References `std::__find_if_not()`, `std::__inplace_stable_partition()`, `std::__stable_partition_adaptive()`, `std::Temporary_buffer<_ForwardIterator, _Tp>::begin()`, `std::Temporary_buffer<_ForwardIterator, _Tp>::requested_size()`, and `std::Temporary_buffer<_ForwardIterator, _Tp>::size()`.

**2.37.2.31** `template<typename _ForwardIterator1, typename _ForwardIterator2> _ForwardIterator2 std::swap_ranges (`  
`_FowardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2 )`

Swap the elements of two sequences.

## Parameters

<code>__first1</code>	A forward iterator.
<code>__last1</code>	A forward iterator.
<code>__first2</code>	A forward iterator.

## Returns

An iterator equal to `first2+(last1-first1)`.

Swaps each element in the range `[first1,last1)` with the corresponding element in the range `[first2,(last1-first1))`. The ranges must not overlap.

Definition at line 165 of file `stl_algobase.h`.

References `std::iter_swap()`.

Referenced by `std::__rotate()`.

**2.37.2.32** `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation> _OutputIterator`  
`std::transform ( _InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op )`

Perform an operation on a sequence.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

<code>__result</code>	An output iterator.
<code>__unary_op</code>	A unary operator.

**Returns**

An output iterator equal to `__result+(__last-__first)`.

Applies the operator to each element in the input range and assigns the results to successive elements of the output sequence. Evaluates `*(__result+N)=unary_op(*(__first+N))` for each `N` in the range `[0,__last-__first)`.

`unary_op` must not alter its argument.

Definition at line 4915 of file `stl_algo.h`.

**2.37.2.33** `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation > _OutputIterator std::transform ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_op )`

Perform an operation on corresponding elements of two sequences.

**Parameters**

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__binary_op</code>	A binary operator.

**Returns**

An output iterator equal to `result+(last-first)`.

Applies the operator to the corresponding elements in the two input ranges and assigns the results to successive elements of the output sequence. Evaluates `*(__result+N)=__binary_op(*(__first1+N),*(__first2+N))` for each `N` in the range `[0,__last1-__first1)`.

`binary_op` must not alter either of its arguments.

Definition at line 4952 of file `stl_algo.h`.

**2.37.2.34** `template<typename _ForwardIterator > _ForwardIterator std::unique ( _ForwardIterator __first, _ForwardIterator __last )`

Remove consecutive duplicate values from a sequence.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.

**Returns**

An iterator designating the end of the resulting sequence.

Removes all but the first element from each group of consecutive values that compare equal. `unique()` is stable, so the relative order of elements that are not removed is unchanged. Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

Definition at line 1174 of file `stl_algo.h`.

References `std::adjacent_find()`.

```
2.37.2.35  template<typename _ForwardIterator, typename _BinaryPredicate> _ForwardIterator std::unique ( _ForwardIterator  
            __first, _ForwardIterator __last, _BinaryPredicate __binary_pred )
```

Remove consecutive values from a sequence using a predicate.



## Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__binary_pred</code>	A binary predicate.

## Returns

An iterator designating the end of the resulting sequence.

Removes all but the first element from each group of consecutive values for which `__binary_pred` returns true. `unique()` is stable, so the relative order of elements that are not removed is unchanged. Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

Definition at line 1214 of file `stl_algo.h`.

References `std::adjacent_find()`.

2.37.2.36 `template<typename _InputIterator, typename _OutputIterator> _OutputIterator std::unique_copy ( _InputIterator __first, _InputIterator __last, _OutputIterator __result ) [inline]`

Copy a sequence, removing consecutive duplicate values.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.

## Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first,__last)` to the range beginning at `__result`, except that only the first element is copied from groups of consecutive elements that compare equal. `unique_copy()` is stable, so the relative order of elements that are copied is unchanged.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 241. Does `unique_copy()` require CopyConstructible and Assignable?

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 538. 241 again: Does `unique_copy()` require CopyConstructible and Assignable?

Definition at line 5116 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__unique_copy()`.

2.37.2.37 `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate> _OutputIterator std::unique_copy ( _InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred ) [inline]`

Copy a sequence, removing consecutive values using a predicate.

## Parameters

<code>__first</code>	An input iterator.
----------------------	--------------------

<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__binary_pred</code>	A binary predicate.

#### Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first,__last)` to the range beginning at `__result`, except that only the first element is copied from groups of consecutive elements for which `__binary_pred` returns true. `unique_copy()` is stable, so the relative order of elements that are copied is unchanged.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 241. Does `unique_copy()` require `CopyConstructible` and `Assignable`?

Definition at line 5156 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__unique_copy()`.

## 2.38 Negators

Collaboration diagram for Negators:



### Classes

- class `std::binary_negate<_Predicate>`
- class `std::unary_negate<_Predicate>`

### Functions

- template<typename \_Predicate>  
`unary_negate<_Predicate>` `std::not1` (const \_Predicate &\_\_pred)
- template<typename \_Predicate>  
`binary_negate<_Predicate>` `std::not2` (const \_Predicate &\_\_pred)

#### 2.38.1 Detailed Description

The functions `not1` and `not2` each take a predicate functor and return an instance of `unary_negate` or `binary_negate`, respectively. These classes are functors whose `operator()` performs the stored predicate function and then returns the negation of the result.

For example, given a vector of integers and a trivial predicate,

```

struct IntGreaterThanThree
: public std::unary_function<int, bool>
{
    bool operator() (int x) { return x > 3; }
};

std::find_if (v.begin(), v.end(), not1(IntGreaterThanThree()));
  
```

The call to `find_if` will locate the first index (*i*) of *v* for which `!(v[i] > 3)` is true.

The `not1/unary_negate` combination works on predicates taking a single argument. The `not2/binary_negate` combination works on predicates which take two arguments.

#### 2.38.2 Function Documentation

**2.38.2.1** template<typename \_Predicate> `unary_negate<_Predicate>` `std::not1` ( const \_Predicate &\_\_pred ) [inline]

One of the [negation functors](#).

Definition at line 369 of file `stl_function.h`.

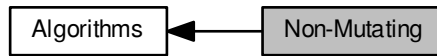
2.38.2.2 `template<typename _Predicate > binary_negate<_Predicate> std::not2 ( const _Predicate & __pred ) [inline]`

One of the [negation functors](#).

Definition at line 394 of file `stl_function.h`.

## 2.39 Non-Mutating

Collaboration diagram for Non-Mutating:



### Functions

- `template<typename _ForwardIterator >`  
`_ForwardIterator std::adjacent\_find (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator std::adjacent\_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __↵  
binary_pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool std::all\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool std::any\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Tp >`  
`iterator_traits< _InputIterator >::difference_type std::count (_InputIterator __first, _InputIterator __last, const _Tp  
& __value)`
- `template<typename _InputIterator, typename _Predicate >`  
`iterator_traits< _InputIterator >::difference_type std::count\_if (_InputIterator __first, _InputIterator __last, _↵  
Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`bool std::equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _II1, typename _II2 >`  
`bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _InputIterator, typename _Tp >`  
`_InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp & __val)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1 std::find\_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __↵  
first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`_ForwardIterator1 std::find\_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __↵  
first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _ForwardIterator >`  
`_InputIterator std::find\_first\_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _↵  
ForwardIterator __last2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`  
`_InputIterator std::find\_first\_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _↵  
ForwardIterator __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator std::find\_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator std::find\_if\_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`

- `template<typename _InputIterator, typename _Function >`  
`_Function std::for_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool std::none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`  
`_ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`

### 2.39.1 Detailed Description

### 2.39.2 Function Documentation

#### 2.39.2.1 `template<typename _ForwardIterator > _ForwardIterator std::adjacent_find ( _ForwardIterator __first, _ForwardIterator __last )`

Find two adjacent values in a sequence that are equal.

##### Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.

##### Returns

The first iterator `i` such that `i` and `i+1` are both valid iterators in `[__first, __last)` and such that `*i == *(i+1)`, or `__last` if no such iterator exists.

Definition at line 4558 of file `stl_algo.h`.

#### 2.39.2.2 `template<typename _ForwardIterator, typename _BinaryPredicate > _ForwardIterator std::adjacent_find ( _ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred )`

Find two adjacent values in a sequence using a predicate.

## Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__binary_pred</code>	A binary predicate.

## Returns

The first iterator `i` such that `i` and `i+1` are both valid iterators in `[__first,__last)` and such that `__binary_pred(*i,*(i+1))` is true, or `__last` if no such iterator exists.

Definition at line 4590 of file `stl_algo.h`.

Referenced by `std::unique()`.

**2.39.2.3** `template<typename _InputIterator, typename _Predicate> bool std::all_of ( _InputIterator __first, _InputIterator __last, _Predicate __pred ) [inline]`

Checks that a predicate is true for all the elements of a sequence.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

## Returns

True if the check is true, false otherwise.

Returns true if `__pred` is true for each element in the range `[__first,__last)`, and false otherwise.

Definition at line 729 of file `stl_algo.h`.

References `std::find_if_not()`.

**2.39.2.4** `template<typename _InputIterator, typename _Predicate> bool std::any_of ( _InputIterator __first, _InputIterator __last, _Predicate __pred ) [inline]`

Checks that a predicate is false for at least an element of a sequence.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

## Returns

True if the check is true, false otherwise.

Returns true if an element exists in the range `[__first,__last)` such that `__pred` is true, and false otherwise.

Definition at line 764 of file `stl_algo.h`.

References `std::none_of()`.

**2.39.2.5** `template<typename _InputIterator, typename _Tp> iterator_traits<_InputIterator>::difference_type std::count ( _InputIterator __first, _InputIterator __last, const _Tp & __value )`

Count the number of copies of a value in a sequence.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__value</code>	The value to be counted.

## Returns

The number of iterators `i` in the range `[__first,__last)` for which `*i == __value`

Definition at line 4622 of file `stl_algo.h`.

Referenced by `std::is_permutation()`.

**2.39.2.6** `template<typename _InputIterator, typename _Predicate> iterator_traits<_InputIterator>::difference_type std::count_if (`  
`_InputIterator __first, _InputIterator __last, _Predicate __pred )`

Count the elements of a sequence for which a predicate is true.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

## Returns

The number of iterators `i` in the range `[__first,__last)` for which `__pred(*i)` is true.

Definition at line 4647 of file `stl_algo.h`.

Referenced by `std::is_permutation()`.

**2.39.2.7** `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate> bool std::equal ( _Iter1 __first1, _Iter1`  
`__last1, _Iter2 __first2, _BinaryPredicate __binary_pred ) [inline]`

Tests a range for element-wise equality.

## Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__binary_pred</code>	A binary predicate <a href="#">functor</a> .

## Returns

A boolean true or false.

This compares the elements of two ranges using the `binary_pred` parameter, and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

Definition at line 1053 of file `stl_algobase.h`.

**2.39.2.8** `template<typename _II1, typename _II2> bool std::equal ( _II1 __first1, _II1 __last1, _II2 __first2 ) [inline]`

Tests a range for element-wise equality.



**Parameters**

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.

**Returns**

A boolean true or false.

This compares the elements of two ranges using `==` and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

Definition at line 1021 of file `stl_algobase.h`.

Referenced by `std::operator==( )`.

**2.39.2.9** `template<typename _InputIterator, typename _Tp > _InputIterator std::find ( _InputIterator __first, _InputIterator __last, const _Tp & __val ) [inline]`

Find the first occurrence of a value in a sequence.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__val</code>	The value to find.

**Returns**

The first iterator `i` in the range `[__first, __last)` such that `*i == __val`, or `__last` if no such iterator exists.

Definition at line 4432 of file `stl_algo.h`.

References `std::__find()`, and `std::__iterator_category()`.

Referenced by `std::__search_n()`, `std::is_permutation()`, `std::remove()`, `std::search()`, and `std::search_n()`.

**2.39.2.10** `template<typename _ForwardIterator1, typename _ForwardIterator2 > _ForwardIterator1 std::find_end ( _ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2 ) [inline]`

Find last matching subsequence in a sequence.

**Parameters**

<code>__first1</code>	Start of range to search.
<code>__last1</code>	End of range to search.
<code>__first2</code>	Start of sequence to match.
<code>__last2</code>	End of sequence to match.

**Returns**

The last iterator `i` in the range `[__first1, __last1-(__last2-__first2))` such that `*(i+N) == *(__first2+N)` for each `N` in the range `[0, __last2-__first2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[__first2, __last2)` and returns an iterator to the `__first` element of the sub-sequence, or `__last1` if the sub-sequence is not found. The sub-sequence will be the last such subsequence contained in `[__first, __last1)`.

Because the sub-sequence must lie completely within the range  $[\_first1, \_last1)$  it must start at a position less than  $\_last1 - (\_last2 - \_first2)$  where  $\_last2 - \_first2$  is the length of the sub-sequence. This means that the returned iterator  $i$  will be in the range  $[\_first1, \_last1 - (\_last2 - \_first2))$

Definition at line 647 of file `stl_algo.h`.

References `std::iterator_category()`.

```
2.39.2.11 template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate > _ForwardIterator1
std::find_end ( _ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2
__last2, _BinaryPredicate __comp ) [inline]
```

Find last matching subsequence in a sequence using a predicate.

#### Parameters

<code>__first1</code>	Start of range to search.
<code>__last1</code>	End of range to search.
<code>__first2</code>	Start of sequence to match.
<code>__last2</code>	End of sequence to match.
<code>__comp</code>	The predicate to use.

#### Returns

The last iterator  $i$  in the range  $[\_first1, \_last1 - (\_last2 - \_first2))$  such that `predicate(*(i+N), (__first2+N))` is true for each  $N$  in the range  $[0, \_last2 - \_first2)$ , or `__last1` if no such iterator exists.

Searches the range  $[\_first1, \_last1)$  for a sub-sequence that compares equal value-by-value with the sequence given by  $[\_first2, \_last2)$  using `comp` as a predicate and returns an iterator to the first element of the sub-sequence, or `__last1` if the sub-sequence is not found. The sub-sequence will be the last such subsequence contained in  $[\_first1, \_last1)$ .

Because the sub-sequence must lie completely within the range  $[\_first1, \_last1)$  it must start at a position less than  $\_last1 - (\_last2 - \_first2)$  where  $\_last2 - \_first2$  is the length of the sub-sequence. This means that the returned iterator  $i$  will be in the range  $[\_first1, \_last1 - (\_last2 - \_first2))$

Definition at line 695 of file `stl_algo.h`.

References `std::iterator_category()`.

```
2.39.2.12 template<typename _InputIterator, typename _ForwardIterator > _InputIterator std::find_first_of ( _InputIterator __first1,
_InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2 )
```

Find element from a set in a sequence.

#### Parameters

<code>__first1</code>	Start of range to search.
<code>__last1</code>	End of range to search.
<code>__first2</code>	Start of match candidates.
<code>__last2</code>	End of match candidates.

#### Returns

The first iterator  $i$  in the range  $[\_first1, \_last1)$  such that  $*i == *(i2)$  such that  $i2$  is an iterator in  $[\_first2, \_last2)$ , or `__last1` if no such iterator exists.

Searches the range  $[\_first1, \_last1)$  for an element that is equal to some element in the range  $[\_first2, \_last2)$ . If found, returns an iterator in the range  $[\_first1, \_last1)$ , otherwise returns `__last1`.

Definition at line 4486 of file `stl_algo.h`.

```
2.39.2.13 template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate> _InputIterator
std::find_first_of ( _InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2,
    _BinaryPredicate __comp )
```

Find element from a set in a sequence using a predicate.

#### Parameters

<code>__first1</code>	Start of range to search.
<code>__last1</code>	End of range to search.
<code>__first2</code>	Start of match candidates.
<code>__last2</code>	End of match candidates.
<code>__comp</code>	Predicate to use.

#### Returns

The first iterator `i` in the range `[__first1, __last1)` such that `comp(*i, *(i2))` is true and `i2` is an iterator in `[__first2, __last2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for an element that is equal to some element in the range `[__first2, __last2)`. If found, returns an iterator in the range `[__first1, __last1)`, otherwise returns `__last1`.

Definition at line 4527 of file `stl_algo.h`.

```
2.39.2.14 template<typename _InputIterator, typename _Predicate> _InputIterator std::find_if ( _InputIterator __first,
    _InputIterator __last, _Predicate __pred ) [inline]
```

Find the first element in a sequence for which a predicate is true.

#### Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

#### Returns

The first iterator `i` in the range `[__first, __last)` such that `__pred(*i)` is true, or `__last` if no such iterator exists.

Definition at line 4456 of file `stl_algo.h`.

References `std::__find_if()`, and `std::__iterator_category()`.

Referenced by `std::is_permutation()`, `std::none_of()`, and `std::remove_if()`.

```
2.39.2.15 template<typename _InputIterator, typename _Predicate> _InputIterator std::find_if_not ( _InputIterator __first,
    _InputIterator __last, _Predicate __pred ) [inline]
```

Find the first element in a sequence for which a predicate is false.

#### Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

**Returns**

The first iterator `i` in the range `[__first,__last)` such that `__pred(*i)` is false, or `__last` if no such iterator exists.

Definition at line 779 of file `stl_algo.h`.

References `std::__find_if_not()`.

Referenced by `std::all_of()`, and `std::is_partitioned()`.

**2.39.2.16** `template<typename _InputIterator, typename _Function> _Function std::for_each ( _InputIterator __first, _InputIterator __last, _Function __f )`

Apply a function to every element of a sequence.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__f</code>	A unary function object.

**Returns**

`__f` (`std::move(__f)` in C++0x).

Applies the function object `__f` to each element in the range `[first,last)`. `__f` must not modify the order of the sequence. If `__f` has a return value it is ignored.

Definition at line 4411 of file `stl_algo.h`.

**2.39.2.17** `template<typename _ForwardIterator1, typename _ForwardIterator2> bool std::is_permutation ( _ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2 )`

Checks whether a permutation of the second sequence is equal to the first sequence.

**Parameters**

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.

**Returns**

true if there exists a permutation of the elements in the range `[__first2, __first2 + (__last1 - __first1))`, beginning with `ForwardIterator2` begin, such that `equal(__first1, __last1, begin)` returns true; otherwise, returns false.

Definition at line 4271 of file `stl_algo.h`.

References `std::advance()`, `std::count()`, `std::distance()`, and `std::find()`.

**2.39.2.18** `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate> bool std::is_permutation ( _ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred )`

Checks whether a permutation of the second sequence is equal to the first sequence.

## Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__pred</code>	A binary predicate.

## Returns

true if there exists a permutation of the elements in the range `[__first2, __first2 + (__last1 - __first1))`, beginning with `ForwardIterator2` begin, such that `equal(__first1, __last1, __begin, __pred)` returns true; otherwise, returns false.

Definition at line 4317 of file `stl_algo.h`.

References `std::advance()`, `std::count_if()`, `std::distance()`, and `std::find_if()`.

2.39.2.19 `template<typename _InputIterator1, typename _InputIterator2> pair<_InputIterator1, _InputIterator2> std::mismatch ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2 )`

Finds the places in ranges which don't match.

## Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.

## Returns

A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using `==` and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

Definition at line 1160 of file `stl_algobase.h`.

2.39.2.20 `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate> pair<_InputIterator1, _InputIterator2> std::mismatch ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred )`

Finds the places in ranges which don't match.

## Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__binary_pred</code>	A binary predicate <a href="#">functor</a> .

## Returns

A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using the `binary_pred` parameter, and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

Definition at line 1198 of file `stl_algobase.h`.

```
2.39.2.21  template<typename _InputIterator, typename _Predicate> bool std::none_of ( _InputIterator __first, _InputIterator  
        __last, _Predicate __pred ) [inline]
```

Checks that a predicate is false for all the elements of a sequence.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

**Returns**

True if the check is true, false otherwise.

Returns true if `__pred` is false for each element in the range `[__first,__last)`, and false otherwise.

Definition at line 746 of file `stl_algo.h`.

References `std::find_if()`.

Referenced by `std::any_of()`, and `std::is_partitioned()`.

**2.39.2.22** `template<typename _ForwardIterator1, typename _ForwardIterator2 > _ForwardIterator1 std::search ( _ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2 )`

Search a sequence for a matching sub-sequence.

**Parameters**

<code>__first1</code>	A forward iterator.
<code>__last1</code>	A forward iterator.
<code>__first2</code>	A forward iterator.
<code>__last2</code>	A forward iterator.

**Returns**

The first iterator `i` in the range `[__first1,__last1-(__last2-__first2))` such that `*(i+N) == *(__first2+N)` for each `N` in the range `[0,__last2-__first2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1,__last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[__first2,__last2)` and returns an iterator to the first element of the sub-sequence, or `__last1` if the sub-sequence is not found.

Because the sub-sequence must lie completely within the range `[__first1,__last1)` it must start at a position less than `__last1-(__last2-__first2)` where `__last2-__first2` is the length of the sub-sequence.

This means that the returned iterator `i` will be in the range `[__first1,__last1-(__last2-__first2))`

Definition at line 4689 of file `stl_algo.h`.

References `std::find()`.

**2.39.2.23** `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate > _ForwardIterator1 std::search ( _ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate )`

Search a sequence for a matching sub-sequence using a predicate.

**Parameters**


---

<code>__first1</code>	A forward iterator.
<code>__last1</code>	A forward iterator.
<code>__first2</code>	A forward iterator.
<code>__last2</code>	A forward iterator.
<code>__predicate</code>	A binary predicate.

**Returns**

The first iterator `i` in the range `[__first1, __last1 - (__last2 - __first2))` such that `__predicate(*(i+N), *(__first2+N))` is true for each `N` in the range `[0, __last2 - __first2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[__first2, __last2)`, using `__predicate` to determine equality, and returns an iterator to the first element of the sub-sequence, or `__last1` if no such iterator exists.

**See also**

`search(_ForwardIter1, _ForwardIter1, _ForwardIter2, _ForwardIter2)`

Definition at line 4761 of file `stl_algo.h`.

2.39.2.24 `template<typename _ForwardIterator, typename _Integer, typename _Tp> _ForwardIterator std::search_n (`  
`_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp & __val )`

Search a sequence for a number of consecutive values.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__count</code>	The number of consecutive values.
<code>__val</code>	The value to find.

**Returns**

The first iterator `i` in the range `[__first, __last - __count)` such that `*(i+N) == __val` for each `N` in the range `[0, __count)`, or `__last` if no such iterator exists.

Searches the range `[__first, __last)` for `count` consecutive elements equal to `__val`.

Definition at line 4835 of file `stl_algo.h`.

References `std::__iterator_category()`, `std::__search_n()`, and `std::find()`.

2.39.2.25 `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate> _ForwardIterator`  
`std::search_n ( _ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp & __val, _BinaryPredicate`  
`__binary_pred )`

Search a sequence for a number of consecutive values using a predicate.

**Parameters**

<code>__first</code>	A forward iterator.
----------------------	---------------------



<code>__last</code>	A forward iterator.
<code>__count</code>	The number of consecutive values.
<code>__val</code>	The value to find.
<code>__binary_pred</code>	A binary predicate.

#### Returns

The first iterator `i` in the range `[__first, __last-__count)` such that `__binary_pred(*(i+N), __val)` is true for each `N` in the range `[0, __count)`, or `__last` if no such iterator exists.

Searches the range `[__first, __last)` for `__count` consecutive elements for which the predicate returns true.

Definition at line 4873 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__search_n()`.

## 2.40 Normal Distributions

Collaboration diagram for Normal Distributions:



### Classes

- class `std::cauchy_distribution< _RealType >`
- struct `std::cauchy_distribution< _RealType >::param_type`
- class `std::chi_squared_distribution< _RealType >`
- struct `std::chi_squared_distribution< _RealType >::param_type`
- class `std::fisher_f_distribution< _RealType >`
- struct `std::fisher_f_distribution< _RealType >::param_type`
- class `std::gamma_distribution< _RealType >`
- struct `std::gamma_distribution< _RealType >::param_type`
- class `std::lognormal_distribution< _RealType >`
- struct `std::lognormal_distribution< _RealType >::param_type`
- class `std::normal_distribution< _RealType >`
- struct `std::normal_distribution< _RealType >::param_type`
- class `std::student_t_distribution< _RealType >`
- struct `std::student_t_distribution< _RealType >::param_type`

### Functions

- template<typename \_RealType >  
bool `std::operator!=` (const `std::normal_distribution< _RealType >` &\_\_d1, const `std::normal_distribution< _RealType >` &\_\_d2)
- template<typename \_RealType >  
bool `std::operator!=` (const `std::lognormal_distribution< _RealType >` &\_\_d1, const `std::lognormal_distribution< _RealType >` &\_\_d2)
- template<typename \_RealType >  
bool `std::operator!=` (const `std::gamma_distribution< _RealType >` &\_\_d1, const `std::gamma_distribution< _RealType >` &\_\_d2)
- template<typename \_RealType >  
bool `std::operator!=` (const `std::chi_squared_distribution< _RealType >` &\_\_d1, const `std::chi_squared_distribution< _RealType >` &\_\_d2)
- template<typename \_RealType >  
bool `std::operator!=` (const `std::cauchy_distribution< _RealType >` &\_\_d1, const `std::cauchy_distribution< _RealType >` &\_\_d2)
- template<typename \_RealType >  
bool `std::operator!=` (const `std::fisher_f_distribution< _RealType >` &\_\_d1, const `std::fisher_f_distribution< _RealType >` &\_\_d2)

- `template<typename _RealType >`  
`bool std::operator!= (const std::student_t_distribution< _RealType > &__d1, const std::student_t_distribution<`  
`_RealType > &__d2)`
- `template<typename _RealType , typename _CharT , typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::cauchy_distribution< _RealType > &__x)`
- `template<typename _RealType , typename _CharT , typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::`  
`::cauchy_distribution< _RealType > &__x)`

#### 2.40.1 Detailed Description

#### 2.40.2 Function Documentation

**2.40.2.1** `template<typename _RealType > bool std::operator!= ( const std::normal_distribution< _RealType > &__d1, const`  
`std::normal_distribution< _RealType > &__d2 ) [inline]`

Return true if two normal distributions are different.

Definition at line 2283 of file random.h.

**2.40.2.2** `template<typename _RealType > bool std::operator!= ( const std::lognormal_distribution< _RealType > &__d1,`  
`const std::lognormal_distribution< _RealType > &__d2 ) [inline]`

Return true if two lognormal distributions are different.

Definition at line 2487 of file random.h.

**2.40.2.3** `template<typename _RealType > bool std::operator!= ( const std::gamma_distribution< _RealType > &__d1, const`  
`std::gamma_distribution< _RealType > &__d2 ) [inline]`

Return true if two gamma distributions are different.

Definition at line 2707 of file random.h.

**2.40.2.4** `template<typename _RealType > bool std::operator!= ( const std::chi_squared_distribution< _RealType > &__d1,`  
`const std::chi_squared_distribution< _RealType > &__d2 ) [inline]`

Return true if two Chi-squared distributions are different.

Definition at line 2917 of file random.h.

**2.40.2.5** `template<typename _RealType > bool std::operator!= ( const std::cauchy_distribution< _RealType > &__d1, const`  
`std::cauchy_distribution< _RealType > &__d2 ) [inline]`

Return true if two Cauchy distributions have different parameters.

Definition at line 3084 of file random.h.

**2.40.2.6** `template<typename _RealType > bool std::operator!= ( const std::fisher_f_distribution< _RealType > &__d1,`  
`const std::fisher_f_distribution< _RealType > &__d2 ) [inline]`

Return true if two Fisher f distributions are different.

Definition at line 3340 of file random.h.

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

Return true if two Student t distributions are different.

Definition at line 3553 of file random.h.

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

Inserts a cauchy\_distribution random number distribution \_\_x into the output stream \_\_os.

Parameters

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

Returns

The output stream with the state of \_\_x inserted or in an error state.

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

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

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A cauchy_distribution random number generator engine.

Returns

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

## 2.41 Numeric\_arrays

### Classes

- class `std::gslice`
- class `std::gslice_array<_Tp>`
- class `std::indirect_array<_Tp>`
- class `std::mask_array<_Tp>`
- class `std::slice`
- class `std::slice_array<_Tp>`

### Macros

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

### Functions

- `std::gslice::gslice()`
- `std::gslice::gslice(size_t __o, const valarray<size_t> &__l, const valarray<size_t> &__s)`
- `std::gslice::gslice(const gslice &)`
- `std::gslice_array<_Tp>::gslice_array(const gslice_array &)`
- `std::indirect_array<_Tp>::indirect_array(const indirect_array &)`
- `std::mask_array<_Tp>::mask_array(const mask_array &)`
- `std::slice::slice()`
- `std::slice::slice(size_t __o, size_t __d, size_t __s)`
- `std::slice_array<_Tp>::slice_array(const slice_array &)`
- `std::gslice::~gslice()`
- `void std::gslice_array<_Tp>::operator%= (const valarray<_Tp> &) const`
- `void std::mask_array<_Tp>::operator%= (const valarray<_Tp> &) const`
- `void std::indirect_array<_Tp>::operator%= (const valarray<_Tp> &) const`
- `template<class _Dom>`  
`void std::gslice_array<_Tp>::operator%= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom>`  
`void std::mask_array<_Tp>::operator%= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom>`  
`void std::indirect_array<_Tp>::operator%= (const _Expr<_Dom, _Tp> &) const`
- `void std::slice_array<_Tp>::operator%= (const valarray<_Tp> &) const`
- `template<class _Dom>`  
`void std::slice_array<_Tp>::operator%= (const _Expr<_Dom, _Tp> &) const`
- `void std::gslice_array<_Tp>::operator&= (const valarray<_Tp> &) const`
- `void std::mask_array<_Tp>::operator&= (const valarray<_Tp> &) const`
- `void std::indirect_array<_Tp>::operator&= (const valarray<_Tp> &) const`
- `template<class _Dom>`  
`void std::gslice_array<_Tp>::operator&= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom>`  
`void std::mask_array<_Tp>::operator&= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom>`  
`void std::indirect_array<_Tp>::operator&= (const _Expr<_Dom, _Tp> &) const`

- void `std::slice_array<_Tp>::operator&=` (const valarray<\_Tp> &) const
- template<class \_Dom >  
void `std::slice_array<_Tp>::operator&=` (const \_Expr<\_Dom, \_Tp> &) const
- void `std::gslice_array<_Tp>::operator*=` (const valarray<\_Tp> &) const
- void `std::mask_array<_Tp>::operator*=` (const valarray<\_Tp> &) const
- void `std::indirect_array<_Tp>::operator*=` (const valarray<\_Tp> &) const
- template<class \_Dom >  
void `std::gslice_array<_Tp>::operator*=` (const \_Expr<\_Dom, \_Tp> &) const
- template<class \_Dom >  
void `std::mask_array<_Tp>::operator*=` (const \_Expr<\_Dom, \_Tp> &) const
- template<class \_Dom >  
void `std::indirect_array<_Tp>::operator*=` (const \_Expr<\_Dom, \_Tp> &) const
- void `std::slice_array<_Tp>::operator+=` (const valarray<\_Tp> &) const
- template<class \_Dom >  
void `std::slice_array<_Tp>::operator+=` (const \_Expr<\_Dom, \_Tp> &) const
- void `std::gslice_array<_Tp>::operator+=` (const valarray<\_Tp> &) const
- void `std::mask_array<_Tp>::operator+=` (const valarray<\_Tp> &) const
- void `std::indirect_array<_Tp>::operator+=` (const valarray<\_Tp> &) const
- template<class \_Dom >  
void `std::gslice_array<_Tp>::operator+=` (const \_Expr<\_Dom, \_Tp> &) const
- template<class \_Dom >  
void `std::mask_array<_Tp>::operator+=` (const \_Expr<\_Dom, \_Tp> &) const
- template<class \_Dom >  
void `std::indirect_array<_Tp>::operator+=` (const \_Expr<\_Dom, \_Tp> &) const
- void `std::slice_array<_Tp>::operator-=` (const valarray<\_Tp> &) const
- template<class \_Dom >  
void `std::slice_array<_Tp>::operator-=` (const \_Expr<\_Dom, \_Tp> &) const
- void `std::gslice_array<_Tp>::operator-=` (const valarray<\_Tp> &) const
- void `std::mask_array<_Tp>::operator-=` (const valarray<\_Tp> &) const
- void `std::indirect_array<_Tp>::operator-=` (const valarray<\_Tp> &) const
- template<class \_Dom >  
void `std::gslice_array<_Tp>::operator-=` (const \_Expr<\_Dom, \_Tp> &) const
- template<class \_Dom >  
void `std::mask_array<_Tp>::operator-=` (const \_Expr<\_Dom, \_Tp> &) const
- template<class \_Dom >  
void `std::indirect_array<_Tp>::operator-=` (const \_Expr<\_Dom, \_Tp> &) const
- void `std::slice_array<_Tp>::operator/=` (const valarray<\_Tp> &) const
- template<class \_Dom >  
void `std::slice_array<_Tp>::operator/=` (const \_Expr<\_Dom, \_Tp> &) const
- void `std::gslice_array<_Tp>::operator/=` (const valarray<\_Tp> &) const
- void `std::mask_array<_Tp>::operator/=` (const valarray<\_Tp> &) const
- void `std::indirect_array<_Tp>::operator/=` (const valarray<\_Tp> &) const
- template<class \_Dom >  
void `std::gslice_array<_Tp>::operator/=` (const \_Expr<\_Dom, \_Tp> &) const
- template<class \_Dom >  
void `std::mask_array<_Tp>::operator/=` (const \_Expr<\_Dom, \_Tp> &) const
- template<class \_Dom >  
void `std::indirect_array<_Tp>::operator/=` (const \_Expr<\_Dom, \_Tp> &) const
- void `std::slice_array<_Tp>::operator/=<=` (const valarray<\_Tp> &) const
- template<class \_Dom >  
void `std::slice_array<_Tp>::operator/=<=` (const \_Expr<\_Dom, \_Tp> &) const
- void `std::gslice_array<_Tp>::operator<=<=` (const valarray<\_Tp> &) const

- void `std::mask_array< _Tp >::operator<=<=` (const valarray< \_Tp > &) const
- void `std::indirect_array< _Tp >::operator<=<=` (const valarray< \_Tp > &) const
- template<class \_Dom >  
void `std::gslice_array< _Tp >::operator<=<=` (const \_Expr< \_Dom, \_Tp > &) const
- template<class \_Dom >  
void `std::mask_array< _Tp >::operator<=<=` (const \_Expr< \_Dom, \_Tp > &) const
- template<class \_Dom >  
void `std::indirect_array< _Tp >::operator<=<=` (const \_Expr< \_Dom, \_Tp > &) const
- void `std::slice_array< _Tp >::operator<=<=` (const valarray< \_Tp > &) const
- template<class \_Dom >  
void `std::slice_array< _Tp >::operator<=<=` (const \_Expr< \_Dom, \_Tp > &) const
- gslice\_array & `std::gslice_array< _Tp >::operator=` (const gslice\_array &)
- indirect\_array & `std::indirect_array< _Tp >::operator=` (const indirect\_array &)
- mask\_array & `std::mask_array< _Tp >::operator=` (const mask\_array &)
- void `std::gslice_array< _Tp >::operator=` (const valarray< \_Tp > &) const
- void `std::mask_array< _Tp >::operator=` (const valarray< \_Tp > &) const
- void `std::indirect_array< _Tp >::operator=` (const valarray< \_Tp > &) const
- gslice & `std::gslice::operator=` (const gslice &)
- void `std::gslice_array< _Tp >::operator=` (const \_Tp &) const
- void `std::mask_array< _Tp >::operator=` (const \_Tp &) const
- void `std::indirect_array< _Tp >::operator=` (const \_Tp &) const
- template<class \_Dom >  
void `std::gslice_array< _Tp >::operator=` (const \_Expr< \_Dom, \_Tp > &) const
- template<class \_Dom >  
void `std::indirect_array< _Tp >::operator=` (const \_Expr< \_Dom, \_Tp > &) const
- slice\_array & `std::slice_array< _Tp >::operator=` (const slice\_array &)
- void `std::slice_array< _Tp >::operator=` (const valarray< \_Tp > &) const
- void `std::slice_array< _Tp >::operator=` (const \_Tp &) const
- template<class \_Dom >  
void `std::slice_array< _Tp >::operator=` (const \_Expr< \_Dom, \_Tp > &) const
- template<class \_Ex >  
void `std::mask_array< _Tp >::operator=` (const \_Expr< \_Ex, \_Tp > &\_\_e) const
- void `std::gslice_array< _Tp >::operator>>=` (const valarray< \_Tp > &) const
- void `std::mask_array< _Tp >::operator>>=` (const valarray< \_Tp > &) const
- void `std::indirect_array< _Tp >::operator>>=` (const valarray< \_Tp > &) const
- template<class \_Dom >  
void `std::gslice_array< _Tp >::operator>>=` (const \_Expr< \_Dom, \_Tp > &) const
- template<class \_Dom >  
void `std::indirect_array< _Tp >::operator>>=` (const \_Expr< \_Dom, \_Tp > &) const
- template<class \_Dom >  
void `std::mask_array< _Tp >::operator>>=` (const \_Expr< \_Dom, \_Tp > &) const
- void `std::slice_array< _Tp >::operator>>=` (const valarray< \_Tp > &) const
- template<class \_Dom >  
void `std::slice_array< _Tp >::operator>>=` (const \_Expr< \_Dom, \_Tp > &) const
- void `std::gslice_array< _Tp >::operator^=` (const valarray< \_Tp > &) const
- void `std::mask_array< _Tp >::operator^=` (const valarray< \_Tp > &) const
- void `std::indirect_array< _Tp >::operator^=` (const valarray< \_Tp > &) const
- template<class \_Dom >  
void `std::gslice_array< _Tp >::operator^=` (const \_Expr< \_Dom, \_Tp > &) const
- template<class \_Dom >  
void `std::indirect_array< _Tp >::operator^=` (const \_Expr< \_Dom, \_Tp > &) const

- `template<class _Dom >`  
`void std::mask_array<_Tp>::operator^= (const _Expr<_Dom, _Tp> &) const`
- `void std::slice_array<_Tp>::operator^= (const valarray<_Tp> &) const`
- `template<class _Dom >`  
`void std::slice_array<_Tp>::operator^= (const _Expr<_Dom, _Tp> &) const`
- `void std::gslice_array<_Tp>::operator|= (const valarray<_Tp> &) const`
- `void std::mask_array<_Tp>::operator|= (const valarray<_Tp> &) const`
- `void std::indirect_array<_Tp>::operator|= (const valarray<_Tp> &) const`
- `template<class _Dom >`  
`void std::gslice_array<_Tp>::operator|= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom >`  
`void std::mask_array<_Tp>::operator|= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom >`  
`void std::indirect_array<_Tp>::operator|= (const _Expr<_Dom, _Tp> &) const`
- `void std::slice_array<_Tp>::operator|= (const valarray<_Tp> &) const`
- `template<class _Dom >`  
`void std::slice_array<_Tp>::operator|= (const _Expr<_Dom, _Tp> &) const`
- `size_t std::slice::size () const`
- `valarray<size_t> std::gslice::size () const`
- `size_t std::slice::start () const`
- `size_t std::gslice::start () const`
- `size_t std::slice::stride () const`
- `valarray<size_t> std::gslice::stride () const`

#### 2.41.1 Detailed Description

#### 2.41.2 Function Documentation

##### 2.41.2.1 `std::gslice::gslice ( )` [inline]

Construct an empty slice.

Definition at line 149 of file `gslice.h`.

##### 2.41.2.2 `std::gslice::gslice ( size_t __o, const valarray<size_t> & __l, const valarray<size_t> & __s )` [inline]

Construct a slice.

Constructs a slice with as many dimensions as the length of the `l` and `s` arrays.

Parameters

<code>__o</code>	Offset in array of first element.
<code>__l</code>	Array of dimension lengths.
<code>__s</code>	Array of dimension strides between array elements.

Definition at line 153 of file `gslice.h`.

##### 2.41.2.3 `std::gslice::gslice ( const gslice & __g )` [inline]

Copy constructor.

Definition at line 158 of file `gslice.h`.



**2.41.2.4** `template<typename _Tp> std::gslice_array<_Tp>::gslice_array ( const gslice_array<_Tp> & __a )`  
`[inline]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 143 of file gslice\_array.h.

**2.41.2.5** `template<typename _Tp> std::indirect_array<_Tp>::indirect_array ( const indirect_array<_Tp> & __a )`  
`[inline]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 143 of file indirect\_array.h.

**2.41.2.6** `template<typename _Tp> std::mask_array<_Tp>::mask_array ( const mask_array<_Tp> & a )` `[inline]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 139 of file mask\_array.h.

**2.41.2.7** `std::slice::slice ( )` `[inline]`

Construct an empty slice.

Definition at line 90 of file slice\_array.h.

**2.41.2.8** `std::slice::slice ( size_t __o, size_t __d, size_t __s )` `[inline]`

Construct a slice.

Parameters

<code>__o</code>	Offset in array of first element.
<code>__d</code>	Number of elements in slice.
<code>__s</code>	Stride between array elements.

Definition at line 94 of file slice\_array.h.

**2.41.2.9** `template<typename _Tp> std::slice_array<_Tp>::slice_array ( const slice_array<_Tp> & a )` `[inline]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 207 of file slice\_array.h.

**2.41.2.10** `std::gslice::~~gslice ( )` `[inline]`

Destructor.

Definition at line 163 of file gslice.h.

**2.41.2.11** `template<typename _Tp> void std::gslice_array<_Tp>::operator%=( const valarray<_Tp> & __v ) const`  
`[inline]`

Modulo slice elements by corresponding elements of `v`.

Definition at line 202 of file gslice\_array.h.

**2.41.2.12** `template<typename _Tp> void std::mask_array<_Tp>::operator%=( const valarray<_Tp> & __v ) const`  
`[inline]`

Modulo slice elements by corresponding elements of `v`.

Definition at line 192 of file mask\_array.h.

**2.41.2.13** `template<typename _Tp> void std::indirect_array<_Tp>::operator%=( const valarray<_Tp> &__v ) const`  
`[inline]`

Modulo slice elements by corresponding elements of *v*.

Definition at line 196 of file indirect\_array.h.

**2.41.2.14** `template<typename _Tp> void std::slice_array<_Tp>::operator%=( const valarray<_Tp> &__v ) const`  
`[inline]`

Modulo slice elements by corresponding elements of *v*.

Definition at line 258 of file slice\_array.h.

**2.41.2.15** `template<typename _Tp> void std::gslice_array<_Tp>::operator&=( const valarray<_Tp> &__v ) const`  
`[inline]`

Logical and slice elements with corresponding elements of *v*.

Definition at line 206 of file gslice\_array.h.

**2.41.2.16** `template<typename _Tp> void std::mask_array<_Tp>::operator&=( const valarray<_Tp> &__v ) const`  
`[inline]`

Logical and slice elements with corresponding elements of *v*.

Definition at line 196 of file mask\_array.h.

**2.41.2.17** `template<typename _Tp> void std::indirect_array<_Tp>::operator&=( const valarray<_Tp> &__v ) const`  
`[inline]`

Logical and slice elements with corresponding elements of *v*.

Definition at line 200 of file indirect\_array.h.

**2.41.2.18** `template<typename _Tp> void std::slice_array<_Tp>::operator&=( const valarray<_Tp> &__v ) const`  
`[inline]`

Logical and slice elements with corresponding elements of *v*.

Definition at line 262 of file slice\_array.h.

**2.41.2.19** `template<typename _Tp> void std::gslice_array<_Tp>::operator*=( const valarray<_Tp> &__v ) const`  
`[inline]`

Multiply slice elements by corresponding elements of *v*.

Definition at line 200 of file gslice\_array.h.

**2.41.2.20** `template<typename _Tp> void std::mask_array<_Tp>::operator*=( const valarray<_Tp> &__v ) const`  
`[inline]`

Multiply slice elements by corresponding elements of *v*.

Definition at line 190 of file mask\_array.h.

**2.41.2.21** `template<typename _Tp> void std::indirect_array<_Tp>::operator*= ( const valarray<_Tp> &__v ) const`  
`[inline]`

Multiply slice elements by corresponding elements of *v*.

Definition at line 194 of file `indirect_array.h`.

**2.41.2.22** `template<typename _Tp> void std::slice_array<_Tp>::operator*= ( const valarray<_Tp> &__v ) const`  
`[inline]`

Multiply slice elements by corresponding elements of *v*.

Definition at line 256 of file `slice_array.h`.

**2.41.2.23** `template<typename _Tp> void std::gslice_array<_Tp>::operator+= ( const valarray<_Tp> &__v ) const`  
`[inline]`

Add corresponding elements of *v* to slice elements.

Definition at line 203 of file `gslice_array.h`.

**2.41.2.24** `template<typename _Tp> void std::mask_array<_Tp>::operator+= ( const valarray<_Tp> &__v ) const`  
`[inline]`

Add corresponding elements of *v* to slice elements.

Definition at line 193 of file `mask_array.h`.

**2.41.2.25** `template<typename _Tp> void std::indirect_array<_Tp>::operator+= ( const valarray<_Tp> &__v ) const`  
`[inline]`

Add corresponding elements of *v* to slice elements.

Definition at line 197 of file `indirect_array.h`.

**2.41.2.26** `template<typename _Tp> void std::slice_array<_Tp>::operator+= ( const valarray<_Tp> &__v ) const`  
`[inline]`

Add corresponding elements of *v* to slice elements.

Definition at line 259 of file `slice_array.h`.

**2.41.2.27** `template<typename _Tp> void std::gslice_array<_Tp>::operator-= ( const valarray<_Tp> &__v ) const`  
`[inline]`

Subtract corresponding elements of *v* from slice elements.

Definition at line 204 of file `gslice_array.h`.

**2.41.2.28** `template<typename _Tp> void std::mask_array<_Tp>::operator-= ( const valarray<_Tp> &__v ) const`  
`[inline]`

Subtract corresponding elements of *v* from slice elements.

Definition at line 194 of file `mask_array.h`.

**2.41.2.29** `template<typename _Tp> void std::indirect_array<_Tp>::operator-= ( const valarray<_Tp> &__v ) const`  
`[inline]`

Subtract corresponding elements of *v* from slice elements.

Definition at line 198 of file indirect\_array.h.

**2.41.230** `template<typename _Tp> void std::slice_array<_Tp>::operator-= ( const valarray<_Tp> &__v ) const`  
`[inline]`

Subtract corresponding elements of *v* from slice elements.

Definition at line 260 of file slice\_array.h.

**2.41.231** `template<typename _Tp> void std::gslice_array<_Tp>::operator/= ( const valarray<_Tp> &__v ) const`  
`[inline]`

Divide slice elements by corresponding elements of *v*.

Definition at line 201 of file gslice\_array.h.

**2.41.232** `template<typename _Tp> void std::mask_array<_Tp>::operator/= ( const valarray<_Tp> &__v ) const`  
`[inline]`

Divide slice elements by corresponding elements of *v*.

Definition at line 191 of file mask\_array.h.

**2.41.233** `template<typename _Tp> void std::indirect_array<_Tp>::operator/= ( const valarray<_Tp> &__v ) const`  
`[inline]`

Divide slice elements by corresponding elements of *v*.

Definition at line 195 of file indirect\_array.h.

**2.41.234** `template<typename _Tp> void std::slice_array<_Tp>::operator/=( const valarray<_Tp> &__v ) const`  
`[inline]`

Divide slice elements by corresponding elements of *v*.

Definition at line 257 of file slice\_array.h.

**2.41.235** `template<typename _Tp> void std::gslice_array<_Tp>::operator<<= ( const valarray<_Tp> &__v ) const`  
`[inline]`

Left shift slice elements by corresponding elements of *v*.

Definition at line 208 of file gslice\_array.h.

**2.41.236** `template<typename _Tp> void std::mask_array<_Tp>::operator<<= ( const valarray<_Tp> &__v ) const`  
`[inline]`

Left shift slice elements by corresponding elements of *v*.

Definition at line 198 of file mask\_array.h.

**2.41.237** `template<typename _Tp> void std::indirect_array<_Tp>::operator<<= ( const valarray<_Tp> &__v ) const`  
`[inline]`

Left shift slice elements by corresponding elements of *v*.

Definition at line 202 of file indirect\_array.h.

**2.41.2.38** `template<typename _Tp> void std::slice_array<_Tp>::operator<<= ( const valarray<_Tp> &__v ) const [inline]`

Left shift slice elements by corresponding elements of *v*.

Definition at line 264 of file slice\_array.h.

**2.41.2.39** `template<typename _Tp> gslice_array<_Tp> & std::gslice_array<_Tp>::operator= ( const gslice_array<_Tp> &__a ) [inline]`

Assignment operator. Assigns slice elements to corresponding elements of *a*.

Definition at line 148 of file gslice\_array.h.

**2.41.2.40** `template<typename _Tp> indirect_array<_Tp> & std::indirect_array<_Tp>::operator= ( const indirect_array<_Tp> &__a ) [inline]`

Assignment operator. Assigns elements to corresponding elements of *a*.

Definition at line 154 of file indirect\_array.h.

**2.41.2.41** `template<typename _Tp> mask_array<_Tp> & std::mask_array<_Tp>::operator= ( const mask_array<_Tp> &__a ) [inline]`

Assignment operator. Assigns elements to corresponding elements of *a*.

Definition at line 149 of file mask\_array.h.

**2.41.2.42** `template<typename _Tp> void std::gslice_array<_Tp>::operator= ( const valarray<_Tp> &__v ) const [inline]`

Assign slice elements to corresponding elements of *v*.

Definition at line 166 of file gslice\_array.h.

**2.41.2.43** `template<typename _Tp> void std::indirect_array<_Tp>::operator= ( const valarray<_Tp> &__v ) const [inline]`

Assign slice elements to corresponding elements of *v*.

Definition at line 168 of file indirect\_array.h.

**2.41.2.44** `gslice & std::gslice::operator= ( const gslice &__g ) [inline]`

Assignment operator.

Definition at line 170 of file gslice.h.

**2.41.2.45** `template<typename _Tp> void std::gslice_array<_Tp>::operator= ( const _Tp &__t ) const [inline]`

Assign all slice elements to *t*.

Definition at line 158 of file gslice\_array.h.

**2.41.2.46** `template<typename _Tp> void std::mask_array<_Tp>::operator= ( const _Tp &__t ) const [inline]`

Assign all slice elements to *t*.

Definition at line 158 of file mask\_array.h.

2.41.2.47 `template<typename _Tp> void std::indirect_array<_Tp>::operator=( const _Tp & __t ) const [inline]`

Assign all slice elements to *t*.

Definition at line 163 of file indirect\_array.h.

2.41.2.48 `template<typename _Tp> slice_array<_Tp> & std::slice_array<_Tp>::operator=( const slice_array<_Tp> & __a ) [inline]`

Assignment operator. Assigns slice elements to corresponding elements of *a*.

Definition at line 215 of file slice\_array.h.

2.41.2.49 `template<typename _Tp> void std::slice_array<_Tp>::operator=( const valarray<_Tp> & __v ) const [inline]`

Assign slice elements to corresponding elements of *v*.

Definition at line 229 of file slice\_array.h.

2.41.2.50 `template<typename _Tp> void std::slice_array<_Tp>::operator=( const _Tp & __t ) const [inline]`

Assign all slice elements to *t*.

Definition at line 224 of file slice\_array.h.

2.41.2.51 `template<typename _Tp> void std::gslice_array<_Tp>::operator>>=( const valarray<_Tp> & __v ) const [inline]`

Right shift slice elements by corresponding elements of *v*.

Definition at line 209 of file gslice\_array.h.

2.41.2.52 `template<typename _Tp> void std::mask_array<_Tp>::operator>>=( const valarray<_Tp> & __v ) const [inline]`

Right shift slice elements by corresponding elements of *v*.

Definition at line 199 of file mask\_array.h.

2.41.2.53 `template<typename _Tp> void std::indirect_array<_Tp>::operator>>=( const valarray<_Tp> & __v ) const [inline]`

Right shift slice elements by corresponding elements of *v*.

Definition at line 203 of file indirect\_array.h.

2.41.2.54 `template<typename _Tp> void std::slice_array<_Tp>::operator>>=( const valarray<_Tp> & __v ) const [inline]`

Right shift slice elements by corresponding elements of *v*.

Definition at line 265 of file slice\_array.h.

2.41.2.55 `template<typename _Tp> void std::gslice_array<_Tp>::operator^=( const valarray<_Tp> & __v ) const [inline]`

Logical xor slice elements with corresponding elements of *v*.

Definition at line 205 of file gslice\_array.h.

**2.41.2.56** `template<typename _Tp> void std::mask_array<_Tp>::operator^= ( const valarray<_Tp> &__v ) const`  
`[inline]`

Logical xor slice elements with corresponding elements of *v*.

Definition at line 195 of file mask\_array.h.

**2.41.2.57** `template<typename _Tp> void std::indirect_array<_Tp>::operator^= ( const valarray<_Tp> &__v ) const`  
`[inline]`

Logical xor slice elements with corresponding elements of *v*.

Definition at line 199 of file indirect\_array.h.

**2.41.2.58** `template<typename _Tp> void std::slice_array<_Tp>::operator^= ( const valarray<_Tp> &__v ) const`  
`[inline]`

Logical xor slice elements with corresponding elements of *v*.

Definition at line 261 of file slice\_array.h.

**2.41.2.59** `template<typename _Tp> void std::gslice_array<_Tp>::operator|= ( const valarray<_Tp> &__v ) const`  
`[inline]`

Logical or slice elements with corresponding elements of *v*.

Definition at line 207 of file gslice\_array.h.

**2.41.2.60** `template<typename _Tp> void std::mask_array<_Tp>::operator|= ( const valarray<_Tp> &__v ) const`  
`[inline]`

Logical or slice elements with corresponding elements of *v*.

Definition at line 197 of file mask\_array.h.

**2.41.2.61** `template<typename _Tp> void std::indirect_array<_Tp>::operator|= ( const valarray<_Tp> &__v ) const`  
`[inline]`

Logical or slice elements with corresponding elements of *v*.

Definition at line 201 of file indirect\_array.h.

**2.41.2.62** `template<typename _Tp> void std::slice_array<_Tp>::operator|= ( const valarray<_Tp> &__v ) const`  
`[inline]`

Logical or slice elements with corresponding elements of *v*.

Definition at line 263 of file slice\_array.h.

**2.41.2.63** `size_t std::slice::size ( ) const` `[inline]`

Return size of slice.

Definition at line 102 of file slice\_array.h.

**2.41.2.64** `valarray<size_t> std::gslice::size ( ) const` `[inline]`

Return array of sizes of slice dimensions.

Definition at line 139 of file gslice.h.

**2.41.2.65** `size_t std::slice::start ( ) const` `[inline]`

Return array offset of first slice element.

Definition at line 98 of file slice\_array.h.

**2.41.2.66** `size_t std::gslice::start ( ) const` `[inline]`

Return array offset of first slice element.

Definition at line 135 of file gslice.h.

**2.41.2.67** `size_t std::slice::stride ( ) const` `[inline]`

Return array stride of slice.

Definition at line 106 of file slice\_array.h.

**2.41.2.68** `valarray< size_t> std::gslice::stride ( ) const` `[inline]`

Return array of array strides for each dimension.

Definition at line 143 of file gslice.h.



## 2.42 Pointer\_abstractions

### Classes

- struct `std::default_delete<_Tp>`
- struct `std::default_delete<_Tp[]>`
- class `std::enable_shared_from_this<_Tp>`
- struct `std::hash<shared_ptr<_Tp>>`
- struct `std::hash<unique_ptr<_Tp,_Dp>>`
- struct `std::owner_less<_Tp>`
- struct `std::owner_less<shared_ptr<_Tp>>`
- struct `std::owner_less<weak_ptr<_Tp>>`
- struct `std::pointer_traits<_Ptr>`
- struct `std::pointer_traits<_Tp*>`
- class `std::shared_ptr<_Tp>`
- class `std::unique_ptr<_Tp,_Dp>`
- class `std::unique_ptr<_Tp[],_Dp>`
- class `std::weak_ptr<_Tp>`

### Functions

- template<typename \_Tp, typename \_Alloc, typename... \_Args>  
shared\_ptr<\_Tp> `std::allocate_shared` (const \_Alloc &\_\_a, \_Args &&...\_\_args)
- template<typename \_Tp, typename \_Tp1>  
shared\_ptr<\_Tp> `std::const_pointer_cast` (const shared\_ptr<\_Tp1> &\_\_r) noexcept
- template<typename \_Tp, typename \_Tp1>  
shared\_ptr<\_Tp> `std::dynamic_pointer_cast` (const shared\_ptr<\_Tp1> &\_\_r) noexcept
- template<typename \_Del, typename \_Tp, \_Lock\_policy \_Lp>  
\_Del \* `std::get_deleter` (const \_\_shared\_ptr<\_Tp,\_Lp> &\_\_p) noexcept
- template<typename \_Tp, typename... \_Args>  
shared\_ptr<\_Tp> `std::make_shared` (\_Args &&...\_\_args)
- template<typename \_Tp1, typename \_Tp2>  
bool `std::operator!=` (const shared\_ptr<\_Tp1> &\_\_a, const shared\_ptr<\_Tp2> &\_\_b) noexcept
- template<typename \_Tp>  
bool `std::operator!=` (const shared\_ptr<\_Tp> &\_\_a, nullptr\_t) noexcept
- template<typename \_Tp>  
bool `std::operator!=` (nullptr\_t, const shared\_ptr<\_Tp> &\_\_a) noexcept
- template<typename \_Tp, typename \_Dp, typename \_Up, typename \_Ep>  
bool `std::operator!=` (const unique\_ptr<\_Tp,\_Dp> &\_\_x, const unique\_ptr<\_Up,\_Ep> &\_\_y)
- template<typename \_Tp, typename \_Dp>  
bool `std::operator!=` (const unique\_ptr<\_Tp,\_Dp> &\_\_x, nullptr\_t) noexcept
- template<typename \_Tp, typename \_Dp>  
bool `std::operator!=` (nullptr\_t, const unique\_ptr<\_Tp,\_Dp> &\_\_x) noexcept
- template<typename \_Tp1, typename \_Tp2>  
bool `std::operator<` (const shared\_ptr<\_Tp1> &\_\_a, const shared\_ptr<\_Tp2> &\_\_b) noexcept
- template<typename \_Tp>  
bool `std::operator<` (const shared\_ptr<\_Tp> &\_\_a, nullptr\_t) noexcept
- template<typename \_Tp>  
bool `std::operator<` (nullptr\_t, const shared\_ptr<\_Tp> &\_\_a) noexcept
- template<typename \_Tp, typename \_Dp, typename \_Up, typename \_Ep>  
bool `std::operator<` (const unique\_ptr<\_Tp,\_Dp> &\_\_x, const unique\_ptr<\_Up,\_Ep> &\_\_y)

- `template<typename _Tp, typename _Dp >`  
`bool std::operator< (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`  
`bool std::operator< (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>`  
`std::basic_ostream< _Ch, _Tr > & std::operator<< (std::basic_ostream< _Ch, _Tr > &__os, const __shared_ptr< _Tp, _Lp > &__p)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool std::operator<= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`  
`bool std::operator<= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool std::operator<= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool std::operator<= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`  
`bool std::operator<= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`  
`bool std::operator<= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool std::operator== (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`  
`bool std::operator== (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool std::operator== (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`  
`bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<typename _Tp, typename _Dp >`  
`bool std::operator== (nullptr_t, const unique_ptr< _Tp, _Dp > &__x) noexcept`
- `template<typename _Tp1, typename _Tp2 >`  
`bool std::operator> (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`  
`bool std::operator> (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool std::operator> (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`  
`bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`  
`bool std::operator> (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool std::operator>= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`  
`bool std::operator>= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool std::operator>= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`  
`bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`

- `template<typename _Tp, typename _Dp>`  
`bool std::operator>= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Tp1>`  
`shared_ptr< _Tp > std::static_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Tp>`  
`void std::swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept`
- `template<typename _Tp, typename _Dp>`  
`void std::swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y) noexcept`
- `template<typename _Tp>`  
`void std::swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept`

### 2.42.1 Detailed Description

### 2.42.2 Function Documentation

- 2.42.2.1 `template<typename _Tp, typename _Alloc, typename... _Args> shared_ptr<_Tp> std::allocate_shared ( const _Alloc & __a, _Args &&... __args ) [inline]`

Create an object that is owned by a `shared_ptr`.

#### Parameters

<code>__a</code>	An allocator.
<code>__args</code>	Arguments for the <code>_Tp</code> object's constructor.

#### Returns

A `shared_ptr` that owns the newly created object.

#### Exceptions

<i>An</i>	exception thrown from <code>_Alloc::allocate</code> or from the constructor of <code>_Tp</code> .
-----------	---

A copy of `__a` will be used to allocate memory for the `shared_ptr` and the new object.

Definition at line 595 of file `shared_ptr.h`.

- 2.42.2.2 `template<typename _Del, typename _Tp, _Lock_policy _Lp> _Del* std::get_deleter ( const __shared_ptr< _Tp, _Lp > & __p ) [inline], [noexcept]`

### 2.2.3.10 `shared_ptr` `get_deleter` (experimental)

Definition at line 76 of file `shared_ptr.h`.

- 2.42.2.3 `template<typename _Tp, typename... _Args> shared_ptr<_Tp> std::make_shared ( _Args &&... __args ) [inline]`

Create an object that is owned by a `shared_ptr`.

#### Parameters

<code>__args</code>	Arguments for the <code>_Tp</code> object's constructor.
---------------------	--

#### Returns

A `shared_ptr` that owns the newly created object.

## Exceptions

<i>std::bad_alloc</i> , or	an exception thrown from the constructor of <i>_Tp</i> .
----------------------------	--

Definition at line 610 of file shared\_ptr.h.

```
2.42.2.4  template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp> std::basic_ostream<_Ch, _Tr>&  
          std::operator<< ( std::basic_ostream<_Ch, _Tr> &__os, const __shared_ptr<_Tp, _Lp> &__p ) [inline]
```

## 2.2.3.7 shared\_ptr I/O

Definition at line 66 of file shared\_ptr.h.

## 2.43 Poisson Distributions

Collaboration diagram for Poisson Distributions:



### Classes

- class `std::discrete_distribution< _IntType >`
- struct `std::discrete_distribution< _IntType >::param_type`
- class `std::exponential_distribution< _RealType >`
- struct `std::exponential_distribution< _RealType >::param_type`
- class `std::extreme_value_distribution< _RealType >`
- struct `std::extreme_value_distribution< _RealType >::param_type`
- class `std::piecewise_constant_distribution< _RealType >`
- struct `std::piecewise_constant_distribution< _RealType >::param_type`
- class `std::piecewise_linear_distribution< _RealType >`
- struct `std::piecewise_linear_distribution< _RealType >::param_type`
- class `std::poisson_distribution< _IntType >`
- struct `std::poisson_distribution< _IntType >::param_type`
- class `std::weibull_distribution< _RealType >`
- struct `std::weibull_distribution< _RealType >::param_type`

### Functions

- template<typename \_IntType >  
bool `std::operator!=` (const `std::poisson_distribution< _IntType >` &\_\_d1, const `std::poisson_distribution< _IntType >` &\_\_d2)
- template<typename \_RealType >  
bool `std::operator!=` (const `std::exponential_distribution< _RealType >` &\_\_d1, const `std::exponential_distribution< _RealType >` &\_\_d2)
- template<typename \_RealType >  
bool `std::operator!=` (const `std::weibull_distribution< _RealType >` &\_\_d1, const `std::weibull_distribution< _RealType >` &\_\_d2)
- template<typename \_RealType >  
bool `std::operator!=` (const `std::extreme_value_distribution< _RealType >` &\_\_d1, const `std::extreme_value_distribution< _RealType >` &\_\_d2)
- template<typename \_IntType >  
bool `std::operator!=` (const `std::discrete_distribution< _IntType >` &\_\_d1, const `std::discrete_distribution< _IntType >` &\_\_d2)
- template<typename \_RealType >  
bool `std::operator!=` (const `std::piecewise_constant_distribution< _RealType >` &\_\_d1, const `std::piecewise_constant_distribution< _RealType >` &\_\_d2)

- `template<typename _RealType >`  
`bool std::operator!= (const std::piecewise_linear_distribution< _RealType > &__d1, const std::piecewise_linear_distribution< _RealType > &__d2)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::extreme_value_distribution< _RealType > &__x)`

#### 2.43.1 Detailed Description

#### 2.43.2 Function Documentation

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

Return true if two Poisson distributions are different.

Definition at line 4625 of file random.h.

**2.43.2.2** `template<typename _RealType > bool std::operator!= ( const std::exponential_distribution< _RealType > & __d1, const std::exponential_distribution< _RealType > & __d2 ) [inline]`

Return true if two exponential distributions have different parameters.

Definition at line 4803 of file random.h.

**2.43.2.3** `template<typename _RealType > bool std::operator!= ( const std::weibull_distribution< _RealType > & __d1, const std::weibull_distribution< _RealType > & __d2 ) [inline]`

Return true if two Weibull distributions have different parameters.

Definition at line 5006 of file random.h.

**2.43.2.4** `template<typename _RealType > bool std::operator!= ( const std::extreme_value_distribution< _RealType > & __d1, const std::extreme_value_distribution< _RealType > & __d2 ) [inline]`

Return true if two extreme value distributions have different parameters.

Definition at line 5209 of file random.h.

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

Return true if two discrete distributions have different parameters.

Definition at line 5469 of file random.h.

**2.43.2.6** `template<typename _RealType> bool std::operator!=( const std::piecewise_constant_distribution< _RealType> &__d1, const std::piecewise_constant_distribution< _RealType> &__d2 ) [inline]`

Return true if two piecewise constant distributions have different parameters.

Definition at line 5736 of file random.h.

**2.43.2.7** `template<typename _RealType> bool std::operator!=( const std::piecewise_linear_distribution< _RealType> &__d1, const std::piecewise_linear_distribution< _RealType> &__d2 ) [inline]`

Return true if two piecewise linear distributions have different parameters.

Definition at line 6006 of file random.h.

**2.43.2.8** `template<typename _RealType, typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits> & std::operator<< ( std::basic_ostream< _CharT, _Traits> &__os, const std::exponential_distribution< _RealType> &__x )`

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

Parameters

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

Returns

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

**2.43.2.9** `template<typename _RealType, typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits> & std::operator<< ( std::basic_ostream< _CharT, _Traits> &__os, const std::weibull_distribution< _RealType> &__x )`

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

Parameters

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

Returns

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

**2.43.2.10** `template<typename _RealType, typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits> & std::operator<< ( std::basic_ostream< _CharT, _Traits> &__os, const std::extreme_value_distribution< _RealType> &__x )`

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

## Parameters

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

## Returns

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

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

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

## Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>exponential_distribution</code> random number generator engine.

## Returns

The input stream with `__x` extracted or in an error state.

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

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

## Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>weibull_distribution</code> random number generator engine.

## Returns

The input stream with `__x` extracted or in an error state.

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

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

## Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>extreme_value_distribution</code> random number generator engine.

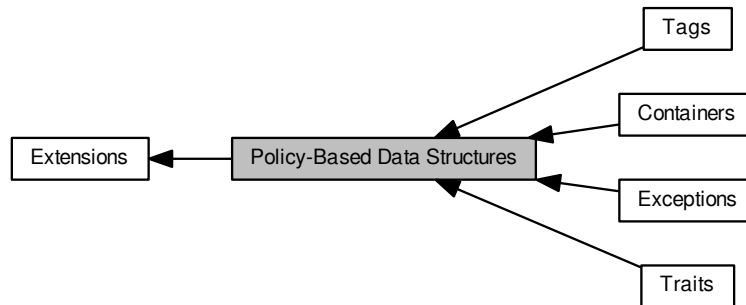
## Returns

The input stream with `__x` extracted or in an error state.



## 2.44 Policy-Based Data Structures

Collaboration diagram for Policy-Based Data Structures:



### Modules

- [Containers](#)
- [Exceptions](#)
- [Tags](#)
- [Traits](#)

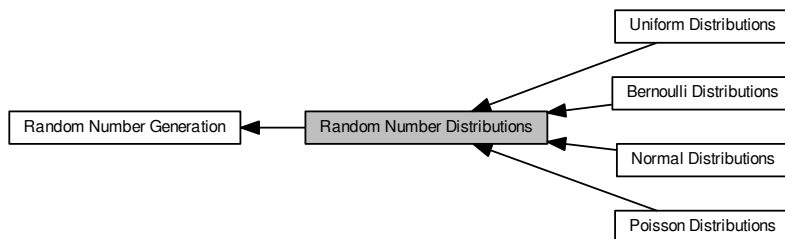
### 2.44.1 Detailed Description

This is a library of policy-based elementary data structures: associative containers and priority queues. It is designed for high-performance, flexibility, semantic safety, and conformance to the corresponding containers in std (except for some points where it differs by design).

For details, see: [http://gcc.gnu.org/onlinedocs/libstdc++/ext/pb\\_ds/index.html](http://gcc.gnu.org/onlinedocs/libstdc++/ext/pb_ds/index.html)

## 2.45 Random Number Distributions

Collaboration diagram for Random Number Distributions:



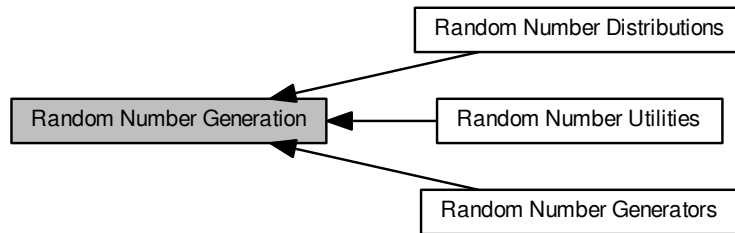
### Modules

- [Bernoulli Distributions](#)
- [Normal Distributions](#)
- [Poisson Distributions](#)
- [Uniform Distributions](#)

### 2.45.1 Detailed Description

## 2.46 Random Number Generation

Collaboration diagram for Random Number Generation:



### Modules

- [Random Number Distributions](#)
- [Random Number Generators](#)
- [Random Number Utilities](#)

### Namespaces

- [std::\\_\\_detail](#)

### Functions

- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator > _RealType std::generate\_canonical (_UniformRandomNumberGenerator &__g)`

#### 2.46.1 Detailed Description

A facility for generating random numbers on selected distributions.

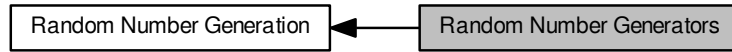
#### 2.46.2 Function Documentation

**2.46.2.1** `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator > _RealType std::generate\_canonical ( _UniformRandomNumberGenerator &__g )`

A function template for converting the output of a (integral) uniform random number generator to a floating point result in the range [0-1).

## 2.47 Random Number Generators

Collaboration diagram for Random Number Generators:



## Classes

- class `std::discard_block_engine< _RandomNumberEngine, __p, __r >`
- class `std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >`
- class `std::linear_congruential_engine< _UIntType, __a, __c, __m >`
- class `std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >`
- class `std::random_device`
- class `std::shuffle_order_engine< _RandomNumberEngine, __k >`

## Typedefs

- typedef `minstd_rand0` **`std::default_random_engine`**
- typedef `shuffle_order_engine< minstd_rand0, 256 >` **`std::knuth_b`**
- typedef `linear_congruential_engine< uint_fast32_t, 48271UL, 0UL, 2147483647UL >` `std::minstd_rand`
- typedef `linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL >` `std::minstd_rand0`
- typedef `mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL >` **`std::mt19937`**
- typedef `mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xffff7eee00000000ULL, 43, 6364136223846793005ULL >` **`std::mt19937_64`**
- typedef `discard_block_engine< ranlux24_base, 223, 23 >` **`std::ranlux24`**
- typedef `subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >` **`std::ranlux24_base`**
- typedef `discard_block_engine< ranlux48_base, 389, 11 >` **`std::ranlux48`**
- typedef `subtract_with_carry_engine< uint_fast64_t, 48, 5, 12 >` **`std::ranlux48_base`**

## Functions

- template<typename \_UIntType, \_UIntType \_\_a, \_UIntType \_\_c, \_UIntType \_\_m>  
bool `std::operator!=` (const `std::linear_congruential_engine< _UIntType, __a, __c, __m >` &\_\_lhs, const `std::linear_congruential_engine< _UIntType, __a, __c, __m >` &\_\_rhs)
- template<typename \_UIntType, size\_t \_\_w, size\_t \_\_n, size\_t \_\_m, size\_t \_\_r, \_UIntType \_\_a, size\_t \_\_u, \_UIntType \_\_d, size\_t \_\_s, \_UIntType \_\_b, size\_t \_\_t, \_UIntType \_\_c, size\_t \_\_l, \_UIntType \_\_f>  
bool `std::operator!=` (const `std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >` &\_\_lhs, const `std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >` &\_\_rhs)

- `template<typename _UIntType, size_t __w, size_t __s, size_t __r>`  
`bool std::operator!= (const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__lhs, const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r>`  
`bool std::operator!= (const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__lhs, const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType >`  
`bool std::operator!= (const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__lhs, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __k>`  
`bool std::operator!= (const std::shuffle_order_engine< _RandomNumberEngine, __k > &__lhs, const std::shuffle_order_engine< _RandomNumberEngine, __k > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > &std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)`

### 2.47.1 Detailed Description

These classes define objects which provide random or pseudorandom numbers, either from a discrete or a continuous interval. The random number generator supplied as a part of this library are all uniform random number generators which provide a sequence of random number uniformly distributed over their range.

A number generator is a function object with an `operator()` that takes zero arguments and returns a number.

A compliant random number generator must satisfy the following requirements.

To be documented.

Table 1: Random Number Generator Requirements

### 2.47.2 Typedef Documentation

#### 2.47.2.1 `typedef linear_congruential_engine<uint_fast32_t, 48271UL, 0UL, 2147483647UL> std::minstd_rand`

An alternative LCR (Lehmer Generator function).

Definition at line 1527 of file `random.h`.

#### 2.47.2.2 `typedef linear_congruential_engine<uint_fast32_t, 16807UL, 0UL, 2147483647UL> std::minstd_rand0`

The classic Minimum Standard `rand0` of Lewis, Goodman, and Miller.

Definition at line 1521 of file `random.h`.

#### 2.47.2.3 `typedef mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL> std::mt19937`

The classic Mersenne Twister.

Reference: M. Matsumoto and T. Nishimura, Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator, ACM Transactions on Modeling and Computer Simulation, Vol. 8, No. 1, January 1998, pp 3-30.

Definition at line 1543 of file `random.h`.

2.47.2.4 `typedef mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xffffeee000000000ULL, 43, 6364136223846793005ULL> std::mt19937_64`

An alternative Mersenne Twister.

Definition at line 1555 of file random.h.

### 2.47.3 Function Documentation

2.47.3.1 `template<typename UIntType, UIntType __a, UIntType __c, UIntType __m> bool std::operator!=( const std::linear_congruential_engine< UIntType, __a, __c, __m > & __lhs, const std::linear_congruential_engine< UIntType, __a, __c, __m > & __rhs ) [inline]`

Compares two linear congruential random number generator objects of the same type for inequality.

Parameters

<code>__lhs</code>	A linear congruential random number generator object.
<code>__rhs</code>	Another linear congruential random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 409 of file random.h.

2.47.3.2 `template<typename UIntType, size_t __w, size_t __n, size_t __m, size_t __r, UIntType __a, size_t __u, UIntType __d, size_t __s, UIntType __b, size_t __t, UIntType __c, size_t __l, UIntType __f> bool std::operator!=( const std::mersenne_twister_engine< UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > & __lhs, const std::mersenne_twister_engine< UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > & __rhs ) [inline]`

Compares two % mersenne\_twister\_engine random number generator objects of the same type for inequality.

Parameters

<code>__lhs</code>	A % mersenne_twister_engine random number generator object.
<code>__rhs</code>	Another % mersenne_twister_engine random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 641 of file random.h.

2.47.3.3 `template<typename UIntType, size_t __w, size_t __s, size_t __r> bool std::operator!=( const std::subtract_with_carry_engine< UIntType, __w, __s, __r > & __lhs, const std::subtract_with_carry_engine< UIntType, __w, __s, __r > & __rhs ) [inline]`

Compares two % subtract\_with\_carry\_engine random number generator objects of the same type for inequality.

Parameters

<code>__lhs</code>	A % subtract_with_carry_engine random number generator object.
<code>__rhs</code>	Another % subtract_with_carry_engine random number generator object.

**Returns**

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 840 of file random.h.

```
2.47.3.4 template<typename _RandomNumberEngine, size_t __p, size_t __r> bool std::operator!= ( const
std::discard_block_engine< _RandomNumberEngine, __p, __r > & __lhs, const std::discard_block_engine<
_RandomNumberEngine, __p, __r > & __rhs ) [inline]
```

Compares two discard\_block\_engine random number generator objects of the same type for inequality.

**Parameters**

<code>__lhs</code>	A discard_block_engine random number generator object.
<code>__rhs</code>	Another discard_block_engine random number generator object.

**Returns**

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 1062 of file random.h.

```
2.47.3.5 template<typename _RandomNumberEngine, size_t __w, typename _UIntType > bool std::operator!=
( const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __lhs, const
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __rhs ) [inline]
```

Compares two independent\_bits\_engine random number generator objects of the same type for inequality.

**Parameters**

<code>__lhs</code>	A independent_bits_engine random number generator object.
<code>__rhs</code>	Another independent_bits_engine random number generator object.

**Returns**

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 1258 of file random.h.

```
2.47.3.6 template<typename _RandomNumberEngine, size_t __k> bool std::operator!= ( const std::shuffle_order_engine<
_RandomNumberEngine, __k > & __lhs, const std::shuffle_order_engine< _RandomNumberEngine, __k > & __rhs )
[inline]
```

Compares two shuffle\_order\_engine random number generator objects of the same type for inequality.

**Parameters**

<code>__lhs</code>	A shuffle_order_engine random number generator object.
--------------------	--

<code>__rhs</code>	Another <code>shuffle_order_engine</code> random number generator object.
--------------------	---

**Returns**

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 1510 of file `random.h`.

**2.47.3.7** `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits  
> std::basic_ostream<_CharT, _Traits>& std::operator<< ( std::basic_ostream<_CharT, _Traits> & __os, const  
std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType> & __x )`

Inserts the current state of a `independent_bits_engine` random number generator engine `__x` into the output stream `__os`.

**Parameters**

<code>__os</code>	An output stream.
<code>__x</code>	A <code>independent_bits_engine</code> random number generator engine.

**Returns**

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

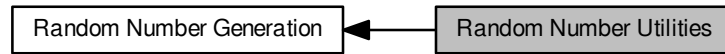
Definition at line 1277 of file `random.h`.

References `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::base()`.



## 2.48 Random Number Utilities

Collaboration diagram for Random Number Utilities:



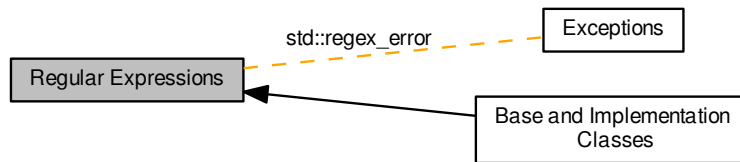
### Classes

- class [std::seed\\_seq](#)

### 2.48.1 Detailed Description

## 2.49 Regular Expressions

Collaboration diagram for Regular Expressions:



### Modules

- [Base and Implementation Classes](#)

### Namespaces

- [std::regex\\_constants](#)

### Classes

- class [std::basic\\_regex< \\_Ch\\_type, \\_Rx\\_traits >](#)
- class [std::match\\_results< \\_Bi\\_iter, \\_Alloc >](#)
- class [std::regex\\_error](#)
- class [std::regex\\_iterator< \\_Bi\\_iter, \\_Ch\\_type, \\_Rx\\_traits >](#)
- class [std::regex\\_token\\_iterator< \\_Bi\\_iter, \\_Ch\\_type, \\_Rx\\_traits >](#)
- struct [std::regex\\_traits< \\_Ch\\_type >](#)
- class [std::sub\\_match< \\_Biter >](#)

### Typedefs

- template<typename \_Bi\_iter, typename \_Ch\_traits, typename \_Ch\_alloc> using **std::\_\_sub\_match\_string** = basic\_string<typename iterator\_traits< \_Bi\_iter >::value\_type, \_Ch\_traits, \_Ch\_alloc>
- typedef match\_results< const char \* > **std::cmatch**
- typedef regex\_iterator< const char \* > **std::cregex\_iterator**
- typedef regex\_token\_iterator< const char \* > [std::cregex\\_token\\_iterator](#)
- typedef sub\_match< const char \* > [std::csub\\_match](#)
- typedef basic\_regex< char > [std::regex](#)
- typedef match\_results< string::const\_iterator > **std::smatch**
- typedef regex\_iterator< string::const\_iterator > **std::sregex\_iterator**
- typedef regex\_token\_iterator< string::const\_iterator > [std::sregex\\_token\\_iterator](#)
- typedef sub\_match< string::const\_iterator > [std::ssub\\_match](#)
- typedef match\_results< const wchar\_t \* > **std::wcmatch**
- typedef regex\_iterator< const wchar\_t \* > **std::wcregex\_iterator**

- typedef regex\_token\_iterator< const wchar\_t \* > [std::wcregex\\_token\\_iterator](#)
- typedef sub\_match< const wchar\_t \* > [std::wcsub\\_match](#)
- typedef basic\_regex< wchar\_t > [std::wregex](#)
- typedef match\_results< wstring::const\_iterator > [std::wsmatch](#)
- typedef regex\_iterator< wstring::const\_iterator > [std::wsregex\\_iterator](#)
- typedef regex\_token\_iterator< wstring::const\_iterator > [std::wsregex\\_token\\_iterator](#)
- typedef sub\_match< wstring::const\_iterator > [std::wssub\\_match](#)

## Functions

- template<typename \_Bi\_iter > const sub\_match< \_Bi\_iter > & [std::\\_\\_unmatched\\_sub](#) ()
- bool [std::regex\\_traits< \\_Ch\\_type >::isctype](#) (\_Ch\_type \_\_c, char\_class\_type \_\_f) const
- template<typename \_Bilter > bool [std::operator!=](#) (const sub\_match< \_Bilter > &\_\_lhs, const sub\_match< \_Bilter > &\_\_rhs)
- template<typename \_Bi\_iter, typename \_Ch\_traits, typename \_Ch\_alloc > bool [std::operator!=](#) (const \_\_sub\_match\_string< \_Bi\_iter, \_Ch\_traits, \_Ch\_alloc > &\_\_lhs, const sub\_match< \_Bi\_iter > &\_\_rhs)
- template<typename \_Bi\_iter, typename \_Ch\_traits, typename \_Ch\_alloc > bool [std::operator!=](#) (const sub\_match< \_Bi\_iter > &\_\_lhs, const \_\_sub\_match\_string< \_Bi\_iter, \_Ch\_traits, \_Ch\_alloc > &\_\_rhs)
- template<typename \_Bi\_iter > bool [std::operator!=](#) (typename iterator\_traits< \_Bi\_iter >::value\_type const \*\_\_lhs, const sub\_match< \_Bi\_iter > &\_\_rhs)
- template<typename \_Bi\_iter > bool [std::operator!=](#) (const sub\_match< \_Bi\_iter > &\_\_lhs, typename iterator\_traits< \_Bi\_iter >::value\_type const \*\_\_rhs)
- template<typename \_Bi\_iter > bool [std::operator!=](#) (typename iterator\_traits< \_Bi\_iter >::value\_type const &\_\_lhs, const sub\_match< \_Bi\_iter > &\_\_rhs)
- template<typename \_Bi\_iter > bool [std::operator!=](#) (const sub\_match< \_Bi\_iter > &\_\_lhs, typename iterator\_traits< \_Bi\_iter >::value\_type const &\_\_rhs)
- template<typename \_Bi\_iter, class \_Alloc > bool [std::operator!=](#) (const match\_results< \_Bi\_iter, \_Alloc > &\_\_m1, const match\_results< \_Bi\_iter, \_Alloc > &\_\_m2)
- template<typename \_Bilter > bool [std::operator<](#) (const sub\_match< \_Bilter > &\_\_lhs, const sub\_match< \_Bilter > &\_\_rhs)
- template<typename \_Bi\_iter, typename \_Ch\_traits, typename \_Ch\_alloc > bool [std::operator<](#) (const \_\_sub\_match\_string< \_Bi\_iter, \_Ch\_traits, \_Ch\_alloc > &\_\_lhs, const sub\_match< \_Bi\_iter > &\_\_rhs)
- template<typename \_Bi\_iter, class \_Ch\_traits, class \_Ch\_alloc > bool [std::operator<](#) (const sub\_match< \_Bi\_iter > &\_\_lhs, const \_\_sub\_match\_string< \_Bi\_iter, \_Ch\_traits, \_Ch\_alloc > &\_\_rhs)
- template<typename \_Bi\_iter > bool [std::operator<](#) (typename iterator\_traits< \_Bi\_iter >::value\_type const \*\_\_lhs, const sub\_match< \_Bi\_iter > &\_\_rhs)
- template<typename \_Bi\_iter > bool [std::operator<](#) (const sub\_match< \_Bi\_iter > &\_\_lhs, typename iterator\_traits< \_Bi\_iter >::value\_type const \*\_\_rhs)
- template<typename \_Bi\_iter > bool [std::operator<](#) (typename iterator\_traits< \_Bi\_iter >::value\_type const &\_\_lhs, const sub\_match< \_Bi\_iter > &\_\_rhs)
- template<typename \_Bi\_iter > bool [std::operator<](#) (const sub\_match< \_Bi\_iter > &\_\_lhs, typename iterator\_traits< \_Bi\_iter >::value\_type const &\_\_rhs)
- template<typename \_Ch\_type, typename \_Ch\_traits, typename \_Bi\_iter > basic\_ostream< \_Ch\_type, \_Ch\_traits > & [std::operator<<](#) (basic\_ostream< \_Ch\_type, \_Ch\_traits > &\_\_os, const sub\_match< \_Bi\_iter > &\_\_m)
- template<typename \_Bilter > bool [std::operator<=](#) (const sub\_match< \_Bilter > &\_\_lhs, const sub\_match< \_Bilter > &\_\_rhs)
- template<typename \_Bi\_iter, typename \_Ch\_traits, typename \_Ch\_alloc > bool [std::operator<=](#) (const \_\_sub\_match\_string< \_Bi\_iter, \_Ch\_traits, \_Ch\_alloc > &\_\_lhs, const sub\_match< \_Bi\_iter > &\_\_rhs)
- template<typename \_Bi\_iter, class \_Ch\_traits, class \_Ch\_alloc > bool [std::operator<=](#) (const sub\_match< \_Bi\_iter > &\_\_lhs, const \_\_sub\_match\_string< \_Bi\_iter, \_Ch\_traits, \_Ch\_alloc > &\_\_rhs)
- template<typename \_Bi\_iter > bool [std::operator<=](#) (typename iterator\_traits< \_Bi\_iter >::value\_type const \*\_\_lhs, const sub\_match< \_Bi\_iter > &\_\_rhs)

- `template<typename _Bi_iter> bool std::operator<= (const sub_match< _Bi_iter> &__lhs, typename iterator_traits< _Bi_iter>::value_type const *__rhs)`
- `template<typename _Bi_iter> bool std::operator<= (typename iterator_traits< _Bi_iter>::value_type const &__lhs, const sub_match< _Bi_iter> &__rhs)`
- `template<typename _Bi_iter> bool std::operator<= (const sub_match< _Bi_iter> &__lhs, typename iterator_traits< _Bi_iter>::value_type const &__rhs)`
- `template<typename _Biter> bool std::operator== (const sub_match< _Biter> &__lhs, const sub_match< _Biter> &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> bool std::operator== (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc> &__lhs, const sub_match< _Bi_iter> &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> bool std::operator== (const sub_match< _Bi_iter> &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc> &__rhs)`
- `template<typename _Bi_iter> bool std::operator== (typename iterator_traits< _Bi_iter>::value_type const *__lhs, const sub_match< _Bi_iter> &__rhs)`
- `template<typename _Bi_iter> bool std::operator== (const sub_match< _Bi_iter> &__lhs, typename iterator_traits< _Bi_iter>::value_type const *__rhs)`
- `template<typename _Bi_iter> bool std::operator== (typename iterator_traits< _Bi_iter>::value_type const &__lhs, const sub_match< _Bi_iter> &__rhs)`
- `template<typename _Bi_iter> bool std::operator== (const sub_match< _Bi_iter> &__lhs, typename iterator_traits< _Bi_iter>::value_type const &__rhs)`
- `template<typename _Bi_iter, typename _Alloc> bool std::operator== (const match_results< _Bi_iter, _Alloc> &__m1, const match_results< _Bi_iter, _Alloc> &__m2)`
- `template<typename _Biter> bool std::operator> (const sub_match< _Biter> &__lhs, const sub_match< _Biter> &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> bool std::operator> (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc> &__lhs, const sub_match< _Bi_iter> &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc> bool std::operator> (const sub_match< _Bi_iter> &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc> &__rhs)`
- `template<typename _Bi_iter> bool std::operator> (typename iterator_traits< _Bi_iter>::value_type const *__lhs, const sub_match< _Bi_iter> &__rhs)`
- `template<typename _Bi_iter> bool std::operator> (const sub_match< _Bi_iter> &__lhs, typename iterator_traits< _Bi_iter>::value_type const *__rhs)`
- `template<typename _Bi_iter> bool std::operator> (typename iterator_traits< _Bi_iter>::value_type const &__lhs, const sub_match< _Bi_iter> &__rhs)`
- `template<typename _Bi_iter> bool std::operator> (const sub_match< _Bi_iter> &__lhs, typename iterator_traits< _Bi_iter>::value_type const &__rhs)`
- `template<typename _Biter> bool std::operator>= (const sub_match< _Biter> &__lhs, const sub_match< _Biter> &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> bool std::operator>= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc> &__lhs, const sub_match< _Bi_iter> &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc> bool std::operator>= (const sub_match< _Bi_iter> &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc> &__rhs)`
- `template<typename _Bi_iter> bool std::operator>= (typename iterator_traits< _Bi_iter>::value_type const *__lhs, const sub_match< _Bi_iter> &__rhs)`
- `template<typename _Bi_iter> bool std::operator>= (const sub_match< _Bi_iter> &__lhs, typename iterator_traits< _Bi_iter>::value_type const *__rhs)`
- `template<typename _Bi_iter> bool std::operator>= (typename iterator_traits< _Bi_iter>::value_type const &__lhs, const sub_match< _Bi_iter> &__rhs)`
- `template<typename _Bi_iter> bool std::operator>= (const sub_match< _Bi_iter> &__lhs, typename iterator_traits< _Bi_iter>::value_type const &__rhs)`
- `template<typename _Ch_type, typename _Rx_traits> void std::swap (basic_regex< _Ch_type, _Rx_traits> &__lhs, basic_regex< _Ch_type, _Rx_traits> &__rhs)`

- `template<typename _Bi_iter, typename _Alloc> void std::swap (match_results< _Bi_iter, _Alloc> &__lhs, match_results< _Bi_iter, _Alloc> &__rhs)`
- `int std::regex\_traits< \_Ch\_type>::value (_Ch_type __ch, int __radix) const`

## Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits> bool std::regex\_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc> &__m, const basic_regex< _Ch_type, _Rx_traits> &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits> bool std::regex\_match (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits> &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Alloc, typename _Rx_traits> bool std::regex\_match (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc> &__m, const basic_regex< _Ch_type, _Rx_traits> &__re, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits> bool std::regex\_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc> &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc>::const_iterator, _Alloc> &__m, const basic_regex< _Ch_type, _Rx_traits> &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Rx_traits> bool std::regex\_match (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits> &__re, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits> bool std::regex\_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator> &__s, const basic_regex< _Ch_type, _Rx_traits> &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits> bool std::regex\_search (_Bi_iter __first, _Bi_iter __last, match_results< _Bi_iter, _Alloc> &__m, const basic_regex< _Ch_type, _Rx_traits> &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits> bool std::regex\_search (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits> &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Alloc, class _Rx_traits> bool std::regex\_search (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc> &__m, const basic_regex< _Ch_type, _Rx_traits> &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Rx_traits> bool std::regex\_search (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits> &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits> bool std::regex\_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator> &__s, const basic_regex< _Ch_type, _Rx_traits> &__e, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits> bool std::regex\_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc> &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc>::const_iterator, _Alloc> &__m, const basic_regex< _Ch_type, _Rx_traits> &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type> _Out_iter std::regex\_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits> &__e, const basic_string< _Ch_type> &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type> basic_string< _Ch_type> std::regex\_replace (const basic_string< _Ch_type> &__s, const basic_regex< _Ch_type, _Rx_traits> &__e, const basic_string< _Ch_type> &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`

## 5.1 Regular Expression Syntax Options

- `enum std::regex\_constants::syntax\_option { \_S\_icode, \_S\_nosubs, \_S\_optimize, \_S\_collate, \_S\_ECMA\_Script, \_S\_basic, \_S\_extended, \_S\_awk, \_S\_grep, \_S\_egrep, \_S\_syntax\_last }`

- typedef unsigned int [std::regex\\_constants::syntax\\_option\\_type](#)
- constexpr syntax\_option\_type [std::regex\\_constants::icase](#)
- constexpr syntax\_option\_type [std::regex\\_constants::nosubs](#)
- constexpr syntax\_option\_type [std::regex\\_constants::optimize](#)
- constexpr syntax\_option\_type [std::regex\\_constants::collate](#)
- constexpr syntax\_option\_type [std::regex\\_constants::ECMAScript](#)
- constexpr syntax\_option\_type [std::regex\\_constants::basic](#)
- constexpr syntax\_option\_type [std::regex\\_constants::extended](#)
- constexpr syntax\_option\_type [std::regex\\_constants::awk](#)
- constexpr syntax\_option\_type [std::regex\\_constants::grep](#)
- constexpr syntax\_option\_type [std::regex\\_constants::egrep](#)

## 5.2 Matching Rules

Matching a regular expression against a sequence of characters [first, last) proceeds according to the rules of the grammar specified for the regular expression object, modified according to the effects listed below for any bitmask elements set.

- enum [std::regex\\_constants::\\_match\\_flag](#) { [\\_S\\_not\\_bol](#), [\\_S\\_not\\_eol](#), [\\_S\\_not\\_bow](#), [\\_S\\_not\\_eow](#), [\\_S\\_any](#), [\\_S\\_not\\_null](#), [\\_S\\_continuous](#), [\\_S\\_prev\\_avail](#), [\\_S\\_sed](#), [\\_S\\_no\\_copy](#), [\\_S\\_first\\_only](#), [\\_S\\_match\\_flag\\_last](#) }
- typedef std::bitset< [\\_S\\_match\\_flag\\_last](#) > [std::regex\\_constants::match\\_flag\\_type](#)
- constexpr match\_flag\_type [std::regex\\_constants::match\\_default](#)
- constexpr match\_flag\_type [std::regex\\_constants::match\\_not\\_bol](#)
- constexpr match\_flag\_type [std::regex\\_constants::match\\_not\\_eol](#)
- constexpr match\_flag\_type [std::regex\\_constants::match\\_not\\_bow](#)
- constexpr match\_flag\_type [std::regex\\_constants::match\\_not\\_eow](#)
- constexpr match\_flag\_type [std::regex\\_constants::match\\_any](#)
- constexpr match\_flag\_type [std::regex\\_constants::match\\_not\\_null](#)
- constexpr match\_flag\_type [std::regex\\_constants::match\\_continuous](#)
- constexpr match\_flag\_type [std::regex\\_constants::match\\_prev\\_avail](#)
- constexpr match\_flag\_type [std::regex\\_constants::format\\_default](#)
- constexpr match\_flag\_type [std::regex\\_constants::format\\_sed](#)
- constexpr match\_flag\_type [std::regex\\_constants::format\\_no\\_copy](#)
- constexpr match\_flag\_type [std::regex\\_constants::format\\_first\\_only](#)

## 5.3 Error Types

- enum [std::regex\\_constants::error\\_type](#) { [\\_S\\_error\\_collate](#), [\\_S\\_error\\_ctype](#), [\\_S\\_error\\_escape](#), [\\_S\\_error\\_backref](#), [\\_S\\_error\\_brack](#), [\\_S\\_error\\_paren](#), [\\_S\\_error\\_brace](#), [\\_S\\_error\\_badbrace](#), [\\_S\\_error\\_range](#), [\\_S\\_error\\_space](#), [\\_S\\_error\\_badrepeat](#), [\\_S\\_error\\_complexity](#), [\\_S\\_error\\_stack](#), [\\_S\\_error\\_last](#) }
- constexpr error\_type [std::regex\\_constants::error\\_collate](#) ( [\\_S\\_error\\_collate](#) )
- constexpr error\_type [std::regex\\_constants::error\\_ctype](#) ( [\\_S\\_error\\_ctype](#) )
- constexpr error\_type [std::regex\\_constants::error\\_escape](#) ( [\\_S\\_error\\_escape](#) )
- constexpr error\_type [std::regex\\_constants::error\\_backref](#) ( [\\_S\\_error\\_backref](#) )
- constexpr error\_type [std::regex\\_constants::error\\_brack](#) ( [\\_S\\_error\\_brack](#) )
- constexpr error\_type [std::regex\\_constants::error\\_paren](#) ( [\\_S\\_error\\_paren](#) )
- constexpr error\_type [std::regex\\_constants::error\\_brace](#) ( [\\_S\\_error\\_brace](#) )
- constexpr error\_type [std::regex\\_constants::error\\_badbrace](#) ( [\\_S\\_error\\_badbrace](#) )
- constexpr error\_type [std::regex\\_constants::error\\_range](#) ( [\\_S\\_error\\_range](#) )
- constexpr error\_type [std::regex\\_constants::error\\_space](#) ( [\\_S\\_error\\_space](#) )
- constexpr error\_type [std::regex\\_constants::error\\_badrepeat](#) ( [\\_S\\_error\\_badrepeat](#) )
- constexpr error\_type [std::regex\\_constants::error\\_complexity](#) ( [\\_S\\_error\\_complexity](#) )
- constexpr error\_type [std::regex\\_constants::error\\_stack](#) ( [\\_S\\_error\\_stack](#) )

### 2.49.1 Detailed Description

A facility for performing regular expression pattern matching.

### 2.49.2 Typedef Documentation

#### 2.49.2.1 `typedef regex_token_iterator<const char*> std::cregex_token_iterator`

Token iterator for C-style NULL-terminated strings.

Definition at line 2469 of file `regex.h`.

#### 2.49.2.2 `typedef sub_match<const char*> std::csub_match`

Standard regex submatch over a C-style null-terminated string.

Definition at line 834 of file `regex.h`.

#### 2.49.2.3 `typedef std::bitset<_S_match_flag_last> std::regex_constants::match_flag_type`

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 198 of file `regex_constants.h`.

#### 2.49.2.4 `typedef basic_regex<char> std::regex`

Standard regular expressions.

Definition at line 706 of file `regex.h`.

#### 2.49.2.5 `typedef regex_token_iterator<string::const_iterator> std::sregex_token_iterator`

Token iterator for standard strings.

Definition at line 2472 of file `regex.h`.

#### 2.49.2.6 `typedef sub_match<string::const_iterator> std::ssub_match`

Standard regex submatch over a standard string.

Definition at line 837 of file `regex.h`.

#### 2.49.2.7 `typedef unsigned int std::regex_constants::syntax_option_type`

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 80 of file `regex_constants.h`.

#### 2.49.2.8 `typedef regex_token_iterator<const wchar_t*> std::wcregex_token_iterator`

Token iterator for C-style NULL-terminated wide strings.

Definition at line 2476 of file regex.h.

#### 2.49.2.9 `typedef sub_match<const wchar_t*> std::wcsub_match`

Regex submatch over a C-style null-terminated wide string.

Definition at line 841 of file regex.h.

#### 2.49.2.10 `typedef basic_regex<wchar_t> std::wregex`

Standard wide-character regular expressions.

Definition at line 710 of file regex.h.

#### 2.49.2.11 `typedef regex_token_iterator<wstring::const_iterator> std::wsregex_token_iterator`

Token iterator for standard wide-character strings.

Definition at line 2479 of file regex.h.

#### 2.49.2.12 `typedef sub_match<wstring::const_iterator> std::wssub_match`

Regex submatch over a standard wide string.

Definition at line 844 of file regex.h.

### 2.49.3 Enumeration Type Documentation

#### 2.49.3.1 `enum std::regex_constants::__match_flag`

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 175 of file regex\_constants.h.

#### 2.49.3.2 `enum std::regex_constants::__syntax_option`

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 54 of file regex\_constants.h.

#### 2.49.3.3 `enum std::regex_constants::error_type`

The expression contained an invalid collating element name.

Definition at line 49 of file regex\_error.h.

### 2.49.4 Function Documentation

#### 2.49.4.1 `constexpr error_type std::regex_constants::error_backref ( _S_error_backref )`

The expression contained an invalid back reference.



#### 2.49.4.2 `constexpr error_type std::regex_constants::error_badbrace ( _S_error_badbrace )`

The expression contained an invalid range in a {} expression.

#### 2.49.4.3 `constexpr error_type std::regex_constants::error_badrepeat ( _S_error_badrepeat )`

One of \*?+{ was not preceded by a valid regular expression.

#### 2.49.4.4 `constexpr error_type std::regex_constants::error_brace ( _S_error_brace )`

The expression contained mismatched { and }

#### 2.49.4.5 `constexpr error_type std::regex_constants::error_brack ( _S_error_brack )`

The expression contained mismatched [ and ].

#### 2.49.4.6 `constexpr error_type std::regex_constants::error_collate ( _S_error_collate )`

The expression contained an invalid collating element name.

#### 2.49.4.7 `constexpr error_type std::regex_constants::error_complexity ( _S_error_complexity )`

The complexity of an attempted match against a regular expression exceeded a pre-set level.

#### 2.49.4.8 `constexpr error_type std::regex_constants::error_ctype ( _S_error_ctype )`

The expression contained an invalid character class name.

#### 2.49.4.9 `constexpr error_type std::regex_constants::error_escape ( _S_error_escape )`

The expression contained an invalid escaped character, or a trailing escape.

#### 2.49.4.10 `constexpr error_type std::regex_constants::error_paren ( _S_error_paren )`

The expression contained mismatched ( and ).

#### 2.49.4.11 `constexpr error_type std::regex_constants::error_range ( _S_error_range )`

The expression contained an invalid character range, such as [b-a] in most encodings.

#### 2.49.4.12 `constexpr error_type std::regex_constants::error_space ( _S_error_space )`

There was insufficient memory to convert the expression into a finite state machine.

#### 2.49.4.13 `constexpr error_type std::regex_constants::error_stack ( _S_error_stack )`

There was insufficient memory to determine whether the regular expression could match the specified character sequence.

#### 2.49.4.14 `template<typename _Ch_type> bool std::regex_traits<_Ch_type>::isctype ( _Ch_type __c, char_class_type __f ) const`

Determines if `c` is a member of an identified class.

## Parameters

<code>__c</code>	a character.
<code>__f</code>	a class type (as returned from <code>lookup_classname</code> ).

## Returns

true if the character `__c` is a member of the classification represented by `__f`, false otherwise.

## Exceptions

<code>std::bad_cast</code>	if the current locale does not have a ctype facet.
----------------------------	--

Definition at line 280 of file `regex.h`.

```
2.49.4.15  template<typename _Bilter > bool std::operator!= ( const sub_match< _Bilter > & __lhs, const sub_match<
    _Bilter > & __rhs )  [inline]
```

Tests the inequivalence of two regular expression submatches.

## Parameters

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

## Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 868 of file `regex.h`.

References `std::sub_match< _Bilter >::compare()`.

```
2.49.4.16  template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc > bool std::operator!= ( const
    __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __lhs, const sub_match< _Bi_iter > & __rhs )
    [inline]
```

Tests the inequivalence of a string and a regular expression submatch.

## Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

## Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 943 of file `regex.h`.

```
2.49.4.17  template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc > bool std::operator!= ( const sub_match<
    _Bi_iter > & __lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __rhs )  [inline]
```

Tests the inequivalence of a regular expression submatch and a string.

## Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

## Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1017 of file `regex.h`.

**2.49.4.18** `template<typename _Bi_iter> bool std::operator!=( typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter > & __rhs ) [inline]`

Tests the inequivalence of an iterator value and a regular expression submatch.

## Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

## Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1091 of file `regex.h`.

**2.49.4.19** `template<typename _Bi_iter> bool std::operator!=( const sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter >::value_type const * __rhs ) [inline]`

Tests the inequivalence of a regular expression submatch and a string.

## Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A pointer to a string.

## Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1165 of file `regex.h`.

**2.49.4.20** `template<typename _Bi_iter> bool std::operator!=( typename iterator_traits< _Bi_iter >::value_type const & __lhs, const sub_match< _Bi_iter > & __rhs ) [inline]`

Tests the inequivalence of a string and a regular expression submatch.

## Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

## Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1242 of file `regex.h`.

```
2.49.4.21  template<typename _Bi_iter> bool std::operator!= ( const sub_match<_Bi_iter> & __lhs, typename iterator_traits<
    _Bi_iter>::value_type const & __rhs )  [inline]
```

Tests the inequivalence of a regular expression submatch and a string.

**Parameters**

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A const string reference.

**Returns**

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1322 of file `regex.h`.

2.49.4.22 `template<typename _Bi_iter, class _Alloc> bool std::operator!=( const match_results< _Bi_iter, _Alloc> & __m1, const match_results< _Bi_iter, _Alloc> & __m2 ) [inline]`

Compares two `match_results` for inequality.

**Returns**

true if the two objects do not refer to the same match, false otherwise.

Definition at line 1834 of file `regex.h`.

2.49.4.23 `template<typename _Bilter> bool std::operator<( const sub_match< _Bilter> & __lhs, const sub_match< _Bilter> & __rhs ) [inline]`

Tests the ordering of two regular expression submatches.

**Parameters**

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

**Returns**

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 879 of file `regex.h`.

References `std::sub_match< _Bilter>::compare()`.

2.49.4.24 `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> bool std::operator<( const sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc> & __lhs, const sub_match< _Bi_iter> & __rhs ) [inline]`

Tests the ordering of a string and a regular expression submatch.

**Parameters**

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

**Returns**

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 955 of file `regex.h`.

References `std::sub_match< _Bilter>::compare()`.

```
2.49.4.25  template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc> bool std::operator< ( const sub_match< _Bi_iter >
            & __lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __rhs ) [inline]
```

Tests the ordering of a regular expression submatch and a string.

**Parameters**

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

**Returns**

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1029 of file `regex.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

**2.49.4.26** `template<typename _Bi_iter> bool std::operator< ( typename iterator_traits<_Bi_iter>::value_type const * __lhs, const sub_match<_Bi_iter> & __rhs ) [inline]`

Tests the ordering of a string and a regular expression submatch.

**Parameters**

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

**Returns**

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1103 of file `regex.h`.

References `std::sub_match<_Biter>::compare()`.

**2.49.4.27** `template<typename _Bi_iter> bool std::operator< ( const sub_match<_Bi_iter> & __lhs, typename iterator_traits<_Bi_iter>::value_type const * __rhs ) [inline]`

Tests the ordering of a regular expression submatch and a string.

**Parameters**

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

**Returns**

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1177 of file `regex.h`.

**2.49.4.28** `template<typename _Bi_iter> bool std::operator< ( typename iterator_traits<_Bi_iter>::value_type const & __lhs, const sub_match<_Bi_iter> & __rhs ) [inline]`

Tests the ordering of a string and a regular expression submatch.

**Parameters**

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

**Returns**

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1254 of file `regex.h`.

References `std::sub_match<_Biter>::compare()`.

**2.49.4.29** `template<typename _Bi_iter> bool std::operator< ( const sub_match<_Bi_iter> & __lhs, typename iterator_traits<_Bi_iter>::value_type const & __rhs ) [inline]`

Tests the ordering of a regular expression submatch and a string.

**Parameters**

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A const string reference.

**Returns**

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1334 of file `regex.h`.

References `std::sub_match<_Biter>::compare()`.

**2.49.4.30** `template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter> basic_ostream<_Ch_type, _Ch_traits> & std::operator<< ( basic_ostream<_Ch_type, _Ch_traits> & __os, const sub_match<_Bi_iter> & __m ) [inline]`

Inserts a matched string into an output stream.

**Parameters**

<code>__os</code>	The output stream.
<code>__m</code>	A submatch string.

**Returns**

the output stream with the submatch string inserted.

Definition at line 1388 of file `regex.h`.

References `std::sub_match<_Biter>::str()`.

**2.49.4.31** `template<typename _Biter> bool std::operator<= ( const sub_match<_Biter> & __lhs, const sub_match<_Biter> & __rhs ) [inline]`

Tests the ordering of two regular expression submatches.



**Parameters**

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

**Returns**

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 890 of file `regex.h`.

References `std::sub_match<_Bilter >::compare()`.

**2.49.4.32** `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc > bool std::operator<= ( const  
__sub_match_string<_Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match<_Bi_iter > &__rhs )  
[inline]`

Tests the ordering of a string and a regular expression submatch.

**Parameters**

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

**Returns**

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 991 of file `regex.h`.

**2.49.4.33** `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc > bool std::operator<= ( const sub_match<_Bi_iter  
> &__lhs, const __sub_match_string<_Bi_iter, _Ch_traits, _Ch_alloc > &__rhs ) [inline]`

Tests the ordering of a regular expression submatch and a string.

**Parameters**

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

**Returns**

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1065 of file `regex.h`.

**2.49.4.34** `template<typename _Bi_iter > bool std::operator<= ( typename iterator_traits<_Bi_iter >::value_type const * __lhs,  
const sub_match<_Bi_iter > &__rhs ) [inline]`

Tests the ordering of a string and a regular expression submatch.

**Parameters**

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

**Returns**

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1139 of file `regex.h`.

**2.49.4.35** `template<typename _Bi_iter> bool std::operator<= ( const sub_match< _Bi_iter> & __lhs, typename iterator_traits< _Bi_iter>::value_type const * __rhs ) [inline]`

Tests the ordering of a regular expression submatch and a string.

**Parameters**

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

**Returns**

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1213 of file `regex.h`.

**2.49.4.36** `template<typename _Bi_iter> bool std::operator<= ( typename iterator_traits< _Bi_iter>::value_type const & __lhs, const sub_match< _Bi_iter> & __rhs ) [inline]`

Tests the ordering of a string and a regular expression submatch.

**Parameters**

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

**Returns**

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1293 of file `regex.h`.

**2.49.4.37** `template<typename _Bi_iter> bool std::operator<= ( const sub_match< _Bi_iter> & __lhs, typename iterator_traits< _Bi_iter>::value_type const & __rhs ) [inline]`

Tests the ordering of a regular expression submatch and a string.

**Parameters**

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A const string reference.

**Returns**

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1373 of file `regex.h`.

2.49.4.38 `template<typename _Bilter > bool std::operator==( const sub_match< _Bilter > & __lhs, const sub_match< _Bilter > & __rhs ) [inline]`

Tests the equivalence of two regular expression submatches.

## Parameters

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

## Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 857 of file `regex.h`.

References `std::sub_match<_Bilter>::compare()`.

**2.49.4.39** `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> bool std::operator==( const  
__sub_match_string<_Bi_iter, _Ch_traits, _Ch_alloc> &__lhs, const sub_match<_Bi_iter> &__rhs )  
[inline]`

Tests the equivalence of a string and a regular expression submatch.

## Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

## Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 930 of file `regex.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`, and `std::sub_match<_Bilter>::compare()`.

**2.49.4.40** `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> bool std::operator==( const sub_match<  
_Bi_iter> &__lhs, const __sub_match_string<_Bi_iter, _Ch_traits, _Ch_alloc> &__rhs ) [inline]`

Tests the equivalence of a regular expression submatch and a string.

## Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

## Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1004 of file `regex.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`, and `std::sub_match<_Bilter>::compare()`.

**2.49.4.41** `template<typename _Bi_iter> bool std::operator==( typename iterator_traits<_Bi_iter>::value_type const * __lhs,  
const sub_match<_Bi_iter> &__rhs ) [inline]`

Tests the equivalence of a C string and a regular expression submatch.

## Parameters

<code>__lhs</code>	A C string.
<code>__rhs</code>	A regular expression submatch.

## Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1078 of file `regex.h`.

References `std::sub_match<_Bilter>::compare()`.

**2.49.4.42** `template<typename _Bi_iter> bool std::operator==( const sub_match<_Bi_iter> & __lhs, typename iterator_traits<_Bi_iter>::value_type const * __rhs ) [inline]`

Tests the equivalence of a regular expression submatch and a string.

## Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A pointer to a string?

## Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1152 of file `regex.h`.

References `std::sub_match<_Bilter>::compare()`.

**2.49.4.43** `template<typename _Bi_iter> bool std::operator==( typename iterator_traits<_Bi_iter>::value_type const & __lhs, const sub_match<_Bi_iter> & __rhs ) [inline]`

Tests the equivalence of a string and a regular expression submatch.

## Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

## Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1226 of file `regex.h`.

References `std::sub_match<_Bilter>::compare()`.

**2.49.4.44** `template<typename _Bi_iter> bool std::operator==( const sub_match<_Bi_iter> & __lhs, typename iterator_traits<_Bi_iter>::value_type const & __rhs ) [inline]`

Tests the equivalence of a regular expression submatch and a string.

## Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A const string reference.

## Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1306 of file `regex.h`.

References `std::sub_match<_Bilter>::compare()`.

**2.49.4.45** `template<typename _Bi_iter, typename _Alloc> bool std::operator==( const match_results<_Bi_iter, _Alloc> & __m1, const match_results<_Bi_iter, _Alloc> & __m2 ) [inline]`

Compares two `match_results` for equality.

## Returns

true if the two objects refer to the same match, false otherwise.

Definition at line 1810 of file `regex.h`.

References `std::match_results<_Bi_iter, _Alloc>::begin()`, `std::match_results<_Bi_iter, _Alloc>::empty()`, `std::match_results<_Bi_iter, _Alloc>::end()`, `std::equal()`, `std::match_results<_Bi_iter, _Alloc>::prefix()`, `std::match_results<_Bi_iter, _Alloc>::ready()`, `std::match_results<_Bi_iter, _Alloc>::size()`, and `std::match_results<_Bi_iter, _Alloc>::suffix()`.

**2.49.4.46** `template<typename _Bilter> bool std::operator> ( const sub_match<_Bilter> & __lhs, const sub_match<_Bilter> & __rhs ) [inline]`

Tests the ordering of two regular expression submatches.

## Parameters

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

## Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 912 of file `regex.h`.

References `std::sub_match<_Bilter>::compare()`.

**2.49.4.47** `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> bool std::operator> ( const __sub_match_string<_Bi_iter, _Ch_traits, _Ch_alloc> & __lhs, const sub_match<_Bi_iter> & __rhs ) [inline]`

Tests the ordering of a string and a regular expression submatch.

## Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

## Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 967 of file `regex.h`.

**2.49.4.48** `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc> bool std::operator> ( const sub_match< _Bi_iter > & __lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __rhs ) [inline]`

Tests the ordering of a regular expression submatch and a string.

## Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

## Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1041 of file `regex.h`.

**2.49.4.49** `template<typename _Bi_iter> bool std::operator> ( typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter > & __rhs ) [inline]`

Tests the ordering of a string and a regular expression submatch.

## Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

## Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1115 of file `regex.h`.

**2.49.4.50** `template<typename _Bi_iter> bool std::operator> ( const sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter >::value_type const * __rhs ) [inline]`

Tests the ordering of a regular expression submatch and a string.

## Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

## Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1189 of file `regex.h`.

```
2.49.4.51  template<typename _Bi_iter > bool std::operator> ( typename iterator_traits<_Bi_iter>::value_type const & __lhs,  
    const sub_match<_Bi_iter> & __rhs ) [inline]
```

Tests the ordering of a string and a regular expression submatch.



## Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

## Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1269 of file `regex.h`.

2.49.4.52 `template<typename _Bi_iter > bool std::operator> ( const sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter >::value_type const & __rhs ) [inline]`

Tests the ordering of a regular expression submatch and a string.

## Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A const string reference.

## Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1349 of file `regex.h`.

2.49.4.53 `template<typename _Bilter > bool std::operator>= ( const sub_match< _Bilter > & __lhs, const sub_match< _Bilter > & __rhs ) [inline]`

Tests the ordering of two regular expression submatches.

## Parameters

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

## Returns

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 901 of file `regex.h`.

References `std::sub_match< _Bilter >::compare()`.

2.49.4.54 `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc > bool std::operator>= ( const sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __lhs, const sub_match< _Bi_iter > & __rhs ) [inline]`

Tests the ordering of a string and a regular expression submatch.

## Parameters

---

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

**Returns**

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 979 of file `regex.h`.

**2.49.4.55** `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc> bool std::operator>= ( const sub_match< _Bi_iter > & __lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc> & __rhs ) [inline]`

Tests the ordering of a regular expression submatch and a string.

**Parameters**

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

**Returns**

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1053 of file `regex.h`.

**2.49.4.56** `template<typename _Bi_iter> bool std::operator>= ( typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter> & __rhs ) [inline]`

Tests the ordering of a string and a regular expression submatch.

**Parameters**

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

**Returns**

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1127 of file `regex.h`.

**2.49.4.57** `template<typename _Bi_iter> bool std::operator>= ( const sub_match< _Bi_iter> & __lhs, typename iterator_traits< _Bi_iter>::value_type const * __rhs ) [inline]`

Tests the ordering of a regular expression submatch and a string.

**Parameters**

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

**Returns**

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1201 of file `regex.h`.

2.49.4.58 `template<typename _Bi_iter> bool std::operator>= ( typename iterator_traits<_Bi_iter>::value_type const & __lhs,  
const sub_match<_Bi_iter> & __rhs ) [inline]`

Tests the ordering of a string and a regular expression submatch.

## Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

## Returns

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1281 of file `regex.h`.

2.49.4.59 `template<typename _Bi_iter > bool std::operator>= ( const sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter >::value_type const & __rhs ) [inline]`

Tests the ordering of a regular expression submatch and a string.

## Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A const string reference.

## Returns

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1361 of file `regex.h`.

2.49.4.60 `template<typename _Bi_iter , typename _Alloc , typename _Ch_type , typename _Rx_traits > bool std::regex_match ( _Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags = regex_constants::match_default )`

Determines if there is a match between the regular expression `e` and all of the character sequence `[first, last)`.

## Parameters

<code>__s</code>	Start of the character sequence to match.
<code>__e</code>	One-past-the-end of the character sequence to match.
<code>__m</code>	The match results.
<code>__re</code>	The regular expression.
<code>__flags</code>	Controls how the regular expression is matched.

## Return values

<code>true</code>	A match exists.
<code>false</code>	Otherwise.

## Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

**Todo** Implement this function.

Definition at line 1878 of file `regex.h`.

Referenced by `std::regex_match()`.

2.49.4.61 `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits > bool std::regex_match ( _Bi_iter __first,  
_Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags  
= regex_constants::match_default )`

Indicates if there is a match between the regular expression  $e$  and all of the character sequence  $[first, last)$ .

## Parameters

<code>__first</code>	Beginning of the character sequence to match.
<code>__last</code>	One-past-the-end of the character sequence to match.
<code>__re</code>	The regular expression.
<code>__flags</code>	Controls how the regular expression is matched.

## Return values

<code>true</code>	A match exists.
<code>false</code>	Otherwise.

## Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 1909 of file `regex.h`.

References `std::regex_match()`.

**2.49.4.62** `template<typename _Ch_type, typename _Alloc, typename _Rx_traits> bool std::regex_match ( const _Ch_type *  
__s, match_results< const _Ch_type *, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __re,  
regex_constants::match_flag_type __f=regex_constants::match_default ) [inline]`

Determines if there is a match between the regular expression `e` and a C-style null-terminated string.

## Parameters

<code>__s</code>	The C-style null-terminated string to match.
<code>__m</code>	The match results.
<code>__re</code>	The regular expression.
<code>__f</code>	Controls how the regular expression is matched.

## Return values

<code>true</code>	A match exists.
<code>false</code>	Otherwise.

## Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 1934 of file `regex.h`.

References `std::regex_match()`.

**2.49.4.63** `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>  
bool std::regex_match ( const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s, match_results< typename  
basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > & __m, const basic_regex< _Ch_type,  
_Rx_traits > & __re, regex_constants::match_flag_type __flags=regex_constants::match_default )  
[inline]`

Determines if there is a match between the regular expression `e` and a string.

## Parameters

<code>__s</code>	The string to match.
<code>__m</code>	The match results.
<code>__re</code>	The regular expression.
<code>__flags</code>	Controls how the regular expression is matched.

**Return values**

<code>true</code>	A match exists.
<code>false</code>	Otherwise.

**Exceptions**

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 1958 of file `regex.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::begin()`, `std::basic_string< _CharT, _Traits, _Alloc >::end()`, and `std::regex_match()`.

```
2.49.4.64  template<typename _Ch_type, class Rx_traits > bool std::regex_match ( const _Ch_type * __s, const basic_regex<
    _Ch_type, Rx_traits > & __re, regex_constants::match_flag_type __f = regex_constants::match_default )
    [inline]
```

Indicates if there is a match between the regular expression `e` and a C-style null-terminated string.

**Parameters**

<code>__s</code>	The C-style null-terminated string to match.
<code>__re</code>	The regular expression.
<code>__f</code>	Controls how the regular expression is matched.

**Return values**

<code>true</code>	A match exists.
<code>false</code>	Otherwise.

**Exceptions**

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 1981 of file `regex.h`.

References `std::regex_match()`.

```
2.49.4.65  template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename Rx_traits >
    bool std::regex_match ( const basic_string< _Ch_type, _Ch_traits, _Str_allocator > & __s, const
    basic_regex< _Ch_type, Rx_traits > & __re, regex_constants::match_flag_type __flags =
    regex_constants::match_default ) [inline]
```

Indicates if there is a match between the regular expression `e` and a string.

**Parameters**

<code>__s</code>	[IN] The string to match.
<code>__re</code>	[IN] The regular expression.

<code>__flags</code>	[IN] Controls how the regular expression is matched.
----------------------	--

## Return values

<code>true</code>	A match exists.
<code>false</code>	Otherwise.

## Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 2003 of file `regex.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::begin()`, `std::basic_string< _CharT, _Traits, _Alloc >::end()`, and `std::regex_match()`.

```
2.49.4.66 template<typename _Out_iter , typename _Bi_iter , typename _Rx_traits , typename _Ch_type > _Out_iter
std::regex_replace ( _Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits
> & __e, const basic_string< _Ch_type > & __fmt, regex_constants::match_flag_type __flags =
regex_constants::match_default ) [inline]
```

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

## Parameters

<code>__out</code>	
<code>__first</code>	
<code>__last</code>	
<code>__e</code>	
<code>__fmt</code>	
<code>__flags</code>	

## Returns

`out`

## Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

**Todo** Implement this function.

Definition at line 2162 of file `regex.h`.

Referenced by `std::regex_replace()`.

```
2.49.4.67 template<typename _Rx_traits , typename _Ch_type > basic_string<_Ch_type> std::regex_replace ( const
basic_string< _Ch_type > & __s, const basic_regex< _Ch_type, _Rx_traits > & __e, const basic_string<
_Ch_type > & __fmt, regex_constants::match_flag_type __flags = regex_constants::match_default )
[inline]
```

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



## Parameters

<code>__s</code>	
<code>__e</code>	
<code>__fmt</code>	
<code>__flags</code>	

## Returns

a copy of string `s` with replacements.

## Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2182 of file `regex.h`.

References `std::back_inserter()`, `std::basic_string<_CharT, _Traits, _Alloc>::begin()`, `std::basic_string<_CharT, _Traits, _Alloc>::end()`, and `std::regex_replace()`.

2.49.4.68 `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits> bool std::regex_search (`  
`_Bi_iter __first, _Bi_iter __last, match_results<_Bi_iter, _Alloc> & __m, const basic_regex<_Ch_type, _Rx_traits`  
`> & __re, regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]`

Searches for a regular expression within a range.

## Parameters

<code>__first</code>	[IN] The start of the string to search.
<code>__last</code>	[IN] One-past-the-end of the string to search.
<code>__m</code>	[OUT] The match results.
<code>__re</code>	[IN] The regular expression to search for.
<code>__flags</code>	[IN] Search policy flags.

## Return values

<i>true</i>	A match was found within the string.
<i>false</i>	No match was found within the string, the content of <code>m</code> is undefined.

## Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

**Todo** Implement this function.

Definition at line 2028 of file `regex.h`.

Referenced by `std::regex_search()`.

2.49.4.69 `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits> bool std::regex_search ( _Bi_iter __first,`  
`_Bi_iter __last, const basic_regex<_Ch_type, _Rx_traits> & __re, regex_constants::match_flag_type __flags`  
`= regex_constants::match_default ) [inline]`

Searches for a regular expression within a range.

## Parameters

<code>__first</code>	[IN] The start of the string to search.
<code>__last</code>	[IN] One-past-the-end of the string to search.
<code>__re</code>	[IN] The regular expression to search for.
<code>__flags</code>	[IN] Search policy flags.

## Return values

<code>true</code>	A match was found within the string.
<code>false</code>	No match was found within the string.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_↵_style.html)

## Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 2049 of file `regex.h`.

References `std::regex_search()`.

2.49.4.70 `template<typename _Ch_type, class _Alloc, class _Rx_traits> bool std::regex_search ( const _Ch_type * __s, match_results< const _Ch_type *, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __f=regex_constants::match_default ) [inline]`

Searches for a regular expression within a C-string.

## Parameters

<code>__s</code>	[IN] A C-string to search for the regex.
<code>__m</code>	[OUT] The set of regex matches.
<code>__e</code>	[IN] The regex to search for in <code>s</code> .
<code>__f</code>	[IN] The search flags.

## Return values

<code>true</code>	A match was found within the string.
<code>false</code>	No match was found within the string, the content of <code>m</code> is undefined.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_↵_style.html)

## Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 2073 of file `regex.h`.

References `std::regex_search()`.

2.49.4.71 `template<typename _Ch_type, typename _Rx_traits> bool std::regex_search ( const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __f=regex_constants::match_default ) [inline]`

Searches for a regular expression within a C-string.

## Parameters

<code>__s</code>	[IN] The C-string to search.
<code>__e</code>	[IN] The regular expression to search for.
<code>__f</code>	[IN] Search policy flags.

## Return values

<code>true</code>	A match was found within the string.
<code>false</code>	No match was found within the string.

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

## Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 2093 of file `regex.h`.

References `std::regex_search()`.

```
2.49.4.72  template<typename _Ch_traits , typename _String_allocator , typename _Ch_type , typename _Rx_traits > bool
           std::regex_search ( const basic_string< _Ch_type, _Ch_traits, _String_allocator > & __s, const basic_regex<
           _Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __flags = regex_constants::match_default
           ) [inline]
```

Searches for a regular expression within a string.

## Parameters

<code>__s</code>	[IN] The string to search.
<code>__e</code>	[IN] The regular expression to search for.
<code>__flags</code>	[IN] Search policy flags.

## Return values

<code>true</code>	A match was found within the string.
<code>false</code>	No match was found within the string.

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

## Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 2113 of file `regex.h`.

References `std::regex_search()`.

```
2.49.4.73  template<typename _Ch_traits , typename _Ch_alloc , typename _Alloc , typename _Ch_type , typename _Rx_traits >
           bool std::regex_search ( const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s, match_results<
           typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > & __m, const basic_regex<
           _Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __f = regex_constants::match_default )
           [inline]
```

Searches for a regular expression within a string.

## Parameters

<code>__s</code>	[IN] A C++ string to search for the regex.
<code>__m</code>	[OUT] The set of regex matches.
<code>__e</code>	[IN] The regex to search for in <code>s</code> .
<code>__f</code>	[IN] The search flags.

## Return values

<code>true</code>	A match was found within the string.
<code>false</code>	No match was found within the string, the content of <code>m</code> is undefined.

## Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 2136 of file `regex.h`.

References `std::basic_string<_CharT, _Traits, _Alloc >::begin()`, `std::basic_string<_CharT, _Traits, _Alloc >::end()`, and `std::regex_search()`.

**2.49.4.74** `template<typename _Ch_type, typename _Rx_traits > void std::swap ( basic_regex<_Ch_type, _Rx_traits > & __lhs, basic_regex<_Ch_type, _Rx_traits > & __rhs ) [inline]`

Swaps the contents of two regular expression objects.

## Parameters

<code>__lhs</code>	First regular expression.
<code>__rhs</code>	Second regular expression.

Definition at line 722 of file `regex.h`.

References `std::basic_regex<_Ch_type, _Rx_traits >::swap()`.

**2.49.4.75** `template<typename _Bi_iter, typename _Alloc > void std::swap ( match_results<_Bi_iter, _Alloc > & __lhs, match_results<_Bi_iter, _Alloc > & __rhs ) [inline]`

Swaps two match results.

## Parameters

<code>__lhs</code>	A match result.
<code>__rhs</code>	A match result.

The contents of the two `match_results` objects are swapped.

Definition at line 1848 of file `regex.h`.

References `std::match_results<_Bi_iter, _Alloc >::swap()`.

**2.49.4.76** `template<typename _Ch_type > int std::regex_traits<_Ch_type >::value ( _Ch_type __ch, int __radix ) const`

Converts a digit to an int.

## Parameters

<code>__ch</code>	a character representing a digit.
-------------------	-----------------------------------

<code>__radix</code>	the radix if the numeric conversion (limited to 8, 10, or 16).
----------------------	--

### Returns

the value represented by the digit `__ch` in base `radix` if the character `__ch` is a valid digit in base `radix`; otherwise returns -1.

Definition at line 314 of file `regex.h`.

References `std::hex()`, and `std::oct()`.

## 2.49.5 Variable Documentation

### 2.49.5.1 `constexpr syntax_option_type std::regex_constants::awk`

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility `awk` in IEEE Std 1003.1-2001. This option is identical to `syntax_option_type` extended, except that C-style escape sequences are supported. These sequences are: `\`, `\a`, `\b`, `\f`, `\n`, `\r`, `\t`, `\v`, `\'`, and `\ddd` (where `ddd` is one, two, or three octal digits).

Definition at line 144 of file `regex_constants.h`.

### 2.49.5.2 `constexpr syntax_option_type std::regex_constants::basic`

Specifies that the grammar recognized by the regular expression engine is that used by POSIX basic regular expressions in IEEE Std 1003.1-2001, Portable Operating System Interface (POSIX), Base Definitions and Headers, Section 9, Regular Expressions [IEEE, Information Technology – Portable Operating System Interface (POSIX), IEEE Standard 1003.1-2001].

Definition at line 126 of file `regex_constants.h`.

### 2.49.5.3 `constexpr syntax_option_type std::regex_constants::collate`

Specifies that character ranges of the form `[a-b]` should be locale sensitive.

Definition at line 107 of file `regex_constants.h`.

### 2.49.5.4 `constexpr syntax_option_type std::regex_constants::ECMAScript`

Specifies that the grammar recognized by the regular expression engine is that used by ECMAScript in ECMA-262 [Ecma International, ECMAScript Language Specification, Standard Ecma-262, third edition, 1999], as modified in section [28.13]. This grammar is similar to that defined in the PERL scripting language but extended with elements found in the POSIX regular expression grammar.

Definition at line 117 of file `regex_constants.h`.

### 2.49.5.5 `constexpr syntax_option_type std::regex_constants::egrep`

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility `grep` when given the `-E` option in IEEE Std 1003.1-2001. This option is identical to `syntax_option_type` extended, except that newlines are treated as whitespace.

Definition at line 160 of file `regex_constants.h`.

### 2.49.5.6 `constexpr syntax_option_type std::regex_constants::extended`

Specifies that the grammar recognized by the regular expression engine is that used by POSIX extended regular expressions in IEEE Std 1003.1-2001, Portable Operating System Interface (POSIX), Base Definitions and Headers, Section 9, Regular Expressions.

Definition at line 134 of file `regex_constants.h`.

#### 2.49.5.7 `constexpr match_flag_type std::regex_constants::format_default`

When a regular expression match is to be replaced by a new string, the new string is constructed using the rules used by the ECMAScript `replace` function in ECMA- 262 [Ecma International, ECMAScript Language Specification, Standard Ecma-262, third edition, 1999], part 15.5.4.11 `String.prototype.replace`. In addition, during search and replace operations all non-overlapping occurrences of the regular expression are located and replaced, and sections of the input that did not match the expression are copied unchanged to the output string.

Format strings (from ECMA-262 [15.5.4.11]):

- `$$` The dollar-sign itself (`$`)
- `$&` The matched substring.
- `$'` The portion of *string* that precedes the matched substring. This would be `match_results::prefix()`.
- `$'` The portion of *string* that follows the matched substring. This would be `match_results::suffix()`.
- `$n` The *n*th capture, where *n* is in `[1,9]` and `$n` is not followed by a decimal digit. If `n <= match_results::size()` and the *n*th capture is undefined, use the empty string instead. If `n > match_results::size()`, the result is implementation-defined.
- `$nn` The *nn*th capture, where *nn* is a two-digit decimal number on `[01, 99]`. If `nn <= match_results::size()` and the *nn*th capture is undefined, use the empty string instead. If `nn > match_results::size()`, the result is implementation-defined.

Definition at line 280 of file `regex_constants.h`.

#### 2.49.5.8 `constexpr match_flag_type std::regex_constants::format_first_only`

When specified during a search and replace operation, only the first occurrence of the regular expression shall be replaced.

Definition at line 301 of file `regex_constants.h`.

#### 2.49.5.9 `constexpr match_flag_type std::regex_constants::format_no_copy`

During a search and replace operation, sections of the character container sequence being searched that do not match the regular expression shall not be copied to the output string.

Definition at line 295 of file `regex_constants.h`.

#### 2.49.5.10 `constexpr match_flag_type std::regex_constants::format_sed`

When a regular expression match is to be replaced by a new string, the new string is constructed using the rules used by the POSIX `sed` utility in IEEE Std 1003.1- 2001 [IEEE, Information Technology – Portable Operating System Interface (POSIX), IEEE Standard 1003.1-2001].

Definition at line 288 of file `regex_constants.h`.

#### 2.49.5.11 `constexpr syntax_option_type std::regex_constants::grep`

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility `grep` in IEEE Std 1003.1-2001. This option is identical to `syntax_option_type basic`, except that newlines are treated as whitespace.

Definition at line 152 of file `regex_constants.h`.

**2.49.5.12 constexpr syntax\_option\_type std::regex\_constants::icase**

Specifies that the matching of regular expressions against a character sequence shall be performed without regard to case.

Definition at line 86 of file `regex_constants.h`.

**2.49.5.13 constexpr match\_flag\_type std::regex\_constants::match\_any**

If more than one match is possible then any match is an acceptable result.

Definition at line 235 of file `regex_constants.h`.

**2.49.5.14 constexpr match\_flag\_type std::regex\_constants::match\_continuous**

The expression only matches a sub-sequence that begins at first .

Definition at line 245 of file `regex_constants.h`.

**2.49.5.15 constexpr match\_flag\_type std::regex\_constants::match\_default**

The default matching rules.

Definition at line 203 of file `regex_constants.h`.

**2.49.5.16 constexpr match\_flag\_type std::regex\_constants::match\_not\_bol**

The first character in the sequence [first, last) is treated as though it is not at the beginning of a line, so the character (^) in the regular expression shall not match [first, first).

Definition at line 210 of file `regex_constants.h`.

**2.49.5.17 constexpr match\_flag\_type std::regex\_constants::match\_not\_bow**

The expression \b is not matched against the sub-sequence [first,first).

Definition at line 223 of file `regex_constants.h`.

**2.49.5.18 constexpr match\_flag\_type std::regex\_constants::match\_not\_eol**

The last character in the sequence [first, last) is treated as though it is not at the end of a line, so the character (\$) in the regular expression shall not match [last, last).

Definition at line 217 of file `regex_constants.h`.

**2.49.5.19 constexpr match\_flag\_type std::regex\_constants::match\_not\_eow**

The expression \b should not be matched against the sub-sequence [last,last).

Definition at line 229 of file `regex_constants.h`.

**2.49.5.20 constexpr match\_flag\_type std::regex\_constants::match\_not\_null**

The expression does not match an empty sequence.

Definition at line 240 of file `regex_constants.h`.

**2.49.5.21 constexpr match\_flag\_type std::regex\_constants::match\_prev\_avail**

—first is a valid iterator position. When this flag is set then the flags `match_not_bol` and `match_not_bow` are ignored by the regular expression algorithms 28.11 and iterators 28.12.

Definition at line 252 of file `regex_constants.h`.

#### 2.49.5.22 `constexpr syntax_option_type std::regex_constants::nosubs`

Specifies that when a regular expression is matched against a character container sequence, no sub-expression matches are to be stored in the supplied `match_results` structure.

Definition at line 93 of file `regex_constants.h`.

#### 2.49.5.23 `constexpr syntax_option_type std::regex_constants::optimize`

Specifies that the regular expression engine should pay more attention to the speed with which regular expressions are matched, and less to the speed with which regular expression objects are constructed. Otherwise it has no detectable effect on the program output.

Definition at line 101 of file `regex_constants.h`.



## 2.50 SGI

Collaboration diagram for SGI:



Because libstdc++ based its implementation of the STL subsections of the library on the SGI 3.3 implementation, we inherited their extensions as well.

They are additionally documented in the [online documentation](#), a copy of which is also shipped with the library source code (in `.../docs/html/documentation.html`). You can also read the documentation [on SGI's site](#), which is still running even though the code is not maintained.

**NB** that the following notes are pulled from various comments all over the place, so they may seem stilted.

## 2.51 Sequences

Collaboration diagram for Sequences:



### Classes

- class `std::basic_string<_CharT, _Traits, _Alloc>`
- class `std::deque<_Tp, _Alloc>`
- class `std::forward_list<_Tp, _Alloc>`
- class `std::list<_Tp, _Alloc>`
- class `std::priority_queue<_Tp, _Sequence, _Compare>`
- class `std::queue<_Tp, _Sequence>`
- class `std::stack<_Tp, _Sequence>`
- class `std::vector<_Tp, _Alloc>`
- class `std::vector<bool, _Alloc>`

### 2.51.1 Detailed Description

Sequences arrange a collection of objects into a strictly linear order.

The differences between sequences are usually due to one or both of the following:

- memory management
- algorithmic complexity

As an example of the first case, `vector` is required to use a contiguous memory layout, while other sequences such as `deque` are not.

The prime reason for choosing one sequence over another should be based on the second category of differences, algorithmic complexity. For example, if you need to perform many inserts and removals from the middle of a sequence, `list` would be ideal. But if you need to perform constant-time access to random elements of the sequence, then `list` should not be used.

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

## 2.52 Set Operation

Collaboration diagram for Set Operation:



### Functions

- `template<typename _InputIterator1, typename _InputIterator2> bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare> bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator> _OutputIterator std::set\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare> _OutputIterator std::set\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator> _OutputIterator std::set\_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare> _OutputIterator std::set\_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator> _OutputIterator std::set\_symmetric\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare> _OutputIterator std::set\_symmetric\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator> _OutputIterator std::set\_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare> _OutputIterator std::set\_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`

### 2.52.1 Detailed Description

These algorithms are common set operations performed on sequences that are already sorted. The number of comparisons will be linear.

## 2.52.2 Function Documentation

2.52.2.1 `template<typename _InputIterator1, typename _InputIterator2 > bool std::includes ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2 )`

Determines whether all elements of a sequence exists in a range.

**Parameters**

<code>__first1</code>	Start of search range.
<code>__last1</code>	End of search range.
<code>__first2</code>	Start of sequence
<code>__last2</code>	End of sequence.

**Returns**

True if each element in `[__first2,__last2)` is contained in order within `[__first1,__last1)`. False otherwise.

This operation expects both `[__first1,__last1)` and `[__first2,__last2)` to be sorted. Searches for the presence of each element in `[__first2,__last2)` within `[__first1,__last1)`. The iterators over each range only move forward, so this is a linear algorithm. If an element in `[__first2,__last2)` is not found before the search iterator reaches `__last2`, false is returned.

Definition at line 3549 of file `stl_algo.h`.

**2.52.2.2** `template<typename _InputIterator1, typename _InputIterator2, typename _Compare > bool std::includes ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp )`

Determines whether all elements of a sequence exists in a range using comparison.

**Parameters**

<code>__first1</code>	Start of search range.
<code>__last1</code>	End of search range.
<code>__first2</code>	Start of sequence
<code>__last2</code>	End of sequence.
<code>__comp</code>	Comparison function to use.

**Returns**

True if each element in `[__first2,__last2)` is contained in order within `[__first1,__last1)` according to `comp`. False otherwise.

This operation expects both `[__first1,__last1)` and `[__first2,__last2)` to be sorted. Searches for the presence of each element in `[__first2,__last2)` within `[__first1,__last1)`, using `comp` to decide. The iterators over each range only move forward, so this is a linear algorithm. If an element in `[__first2,__last2)` is not found before the search iterator reaches `__last2`, false is returned.

Definition at line 3600 of file `stl_algo.h`.

**2.52.2.3** `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator > _OutputIterator std::set_difference ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result )`

Return the difference of two sorted ranges.

**Parameters**

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.

<code>__last2</code>	End of second range.
----------------------	----------------------

**Returns**

End of the output range.

This operation iterates over both ranges, copying elements present in the first range but not the second in order to the output range. Iterators increment for each range. When the current element of the first range is less than the second, that element is copied and the iterator advances. If the current element of the second range is less, the iterator advances, but no element is copied. If an element is contained in both ranges, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5987 of file `stl_algo.h`.

```
2.52.2.4 template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare
> _OutputIterator std::set_difference ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,
    _InputIterator2 __last2, _OutputIterator __result, _Compare __comp )
```

Return the difference of two sorted ranges using comparison functor.

**Parameters**

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__comp</code>	The comparison functor.

**Returns**

End of the output range.

This operation iterates over both ranges, copying elements present in the first range but not the second in order to the output range. Iterators increment for each range. When the current element of the first range is less than the second according to `__comp`, that element is copied and the iterator advances. If the current element of the second range is less, no element is copied and the iterator advances. If an element is contained in both ranges according to `__comp`, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 6048 of file `stl_algo.h`.

```
2.52.2.5 template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator > _OutputIterator
std::set_intersection ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2,
    _OutputIterator __result )
```

Return the intersection of two sorted ranges.

**Parameters**

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.

**Returns**

End of the output range.

This operation iterates over both ranges, copying elements present in both ranges in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that iterator advances. If an element is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5872 of file `stl_algo.h`.

```
2.52.2.6 template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare
> _OutputIterator std::set_intersection ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,
    _InputIterator2 __last2, _OutputIterator __result, _Compare __comp )
```

Return the intersection of two sorted ranges using comparison functor.

**Parameters**

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__comp</code>	The comparison functor.

**Returns**

End of the output range.

This operation iterates over both ranges, copying elements present in both ranges in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to `__comp`, that iterator advances. If an element is contained in both ranges according to `__comp`, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5929 of file `stl_algo.h`.

```
2.52.2.7 template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator > _OutputIterator
std::set_symmetric_difference ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2
    __last2, _OutputIterator __result )
```

Return the symmetric difference of two sorted ranges.

**Parameters**

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.

**Returns**

End of the output range.

This operation iterates over both ranges, copying elements present in one range but not the other in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that element is copied and the iterator advances. If an element is contained in both ranges, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 6106 of file `stl_algo.h`.

```
2.52.2.8 template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >
           _OutputIterator std::set_symmetric_difference ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,
           _InputIterator2 __last2, _OutputIterator __result, _Compare __comp )
```

Return the symmetric difference of two sorted ranges using comparison functor.



**Parameters**

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__comp</code>	The comparison functor.

**Returns**

End of the output range.

This operation iterates over both ranges, copying elements present in one range but not the other in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to `comp`, that element is copied and the iterator advances. If an element is contained in both ranges according to `__comp`, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 6172 of file `stl_algo.h`.

**2.52.2.9** `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator > _OutputIterator std::set_union ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result )`

Return the union of two sorted ranges.

**Parameters**

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.

**Returns**

End of the output range.

This operation iterates over both ranges, copying elements present in each range in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that element is copied and the iterator advanced. If an element is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5738 of file `stl_algo.h`.

**2.52.2.10** `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare > _OutputIterator std::set_union ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp )`

Return the union of two sorted ranges using a comparison functor.

**Parameters**

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.

<code>__last2</code>	End of second range.
<code>__comp</code>	The comparison functor.

**Returns**

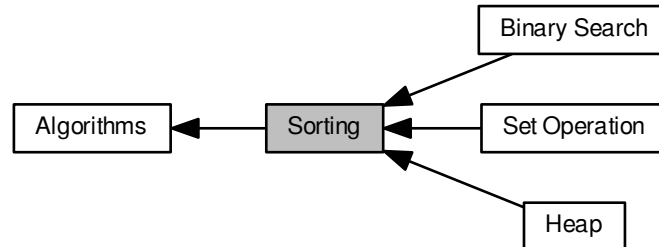
End of the output range.

This operation iterates over both ranges, copying elements present in each range in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to `__comp`, that element is copied and the iterator advanced. If an equivalent element according to `__comp` is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5805 of file `stl_algo.h`.

## 2.53 Sorting

Collaboration diagram for Sorting:



### Modules

- [Binary Search](#)
- [Heap](#)
- [Set Operation](#)

### Functions

- `template<typename _BidirectionalIterator > void std::inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare > void std::inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator > bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare > bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator > _ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare > _ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _I1, typename _I2 > bool std::lexicographical\_compare (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2)`
- `template<typename _I1, typename _I2, typename _Compare > bool std::lexicographical\_compare (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2, _Compare __comp)`
- `template<typename _Tp > const _Tp & std::max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare > const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _ForwardIterator > _ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare > _ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`

- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator > _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare > _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Tp > const _Tp & std::min (const _Tp & __a, const _Tp & __b)`
- `template<typename _Tp, typename _Compare > const _Tp & std::min (const _Tp & __a, const _Tp & __b, _Compare __comp)`
- `template<typename _ForwardIterator > _ForwardIterator std::min\_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare > _ForwardIterator std::min\_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Tp > pair< const _Tp &, const _Tp & > std::minmax (const _Tp & __a, const _Tp & __b)`
- `template<typename _Tp, typename _Compare > pair< const _Tp &, const _Tp & > std::minmax (const _Tp & __a, const _Tp & __b, _Compare __comp)`
- `template<typename _ForwardIterator > pair< _ForwardIterator, _ForwardIterator > std::minmax\_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare > pair< _ForwardIterator, _ForwardIterator > std::minmax\_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _BidirectionalIterator > bool std::next\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare > bool std::next\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator > void std::nth\_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare > void std::nth\_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator > void std::partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare > void std::partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator > _RandomAccessIterator std::partial\_sort\_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare > _RandomAccessIterator std::partial\_sort\_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _BidirectionalIterator > bool std::prev\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare > bool std::prev\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator > void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare > void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator > void std::stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare > void std::stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

### 2.53.1 Detailed Description

### 2.53.2 Function Documentation

**2.53.2.1** `template<typename _BidirectionalIterator > void std::inplace_merge ( _BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last )`

Merges two sorted ranges in place.

#### Parameters

<code>__first</code>	An iterator.
<code>__middle</code>	Another iterator.
<code>__last</code>	Another iterator.

#### Returns

Nothing.

Merges two sorted and consecutive ranges, [`__first`,`__middle`) and [`__middle`,`__last`), and puts the result in [`__first`,`__last`). The output will be sorted. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

If enough additional memory is available, this takes (`__last`-`__first`)-1 comparisons. Otherwise an NlogN algorithm is used, where N is `distance(__first,__last)`.

Definition at line 3154 of file `stl_algo.h`.

References `std::__merge_adaptive()`, `std::__merge_without_buffer()`, `std::Temporary_buffer<_ForwardIterator, _Tp>::begin()`, `std::distance()`, and `std::Temporary_buffer<_ForwardIterator, _Tp>::size()`.

**2.53.2.2** `template<typename _BidirectionalIterator, typename _Compare > void std::inplace_merge ( _BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp )`

Merges two sorted ranges in place.

#### Parameters

<code>__first</code>	An iterator.
<code>__middle</code>	Another iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A functor to use for comparisons.

#### Returns

Nothing.

Merges two sorted and consecutive ranges, [`__first`,`__middle`) and [`__middle`,`__last`), and puts the result in [`__first`,`__last`). The output will be sorted. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

If enough additional memory is available, this takes (`__last`-`__first`)-1 comparisons. Otherwise an NlogN algorithm is used, where N is `distance(__first,__last)`.

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

Definition at line 3209 of file `stl_algo.h`.

References `std::__merge_adaptive()`, `std::__merge_without_buffer()`, `std::Temporary_buffer<_ForwardIterator, _Tp>::begin()`, `std::distance()`, and `std::Temporary_buffer<_ForwardIterator, _Tp>::size()`.

```
2.53.2.3  template<typename _ForwardIterator > bool std::is_sorted ( _ForwardIterator __first, _ForwardIterator __last )  
          [inline]
```

Determines whether the elements of a sequence are sorted.

**Parameters**

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.

**Returns**

True if the elements are sorted, false otherwise.

Definition at line 3952 of file `stl_algo.h`.

References `std::is_sorted_until()`.

**2.53.2.4** `template<typename _ForwardIterator, typename _Compare> bool std::is_sorted ( _ForwardIterator __first, _ForwardIterator __last, _Compare __comp ) [inline]`

Determines whether the elements of a sequence are sorted according to a comparison functor.

**Parameters**

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

**Returns**

True if the elements are sorted, false otherwise.

Definition at line 3966 of file `stl_algo.h`.

References `std::is_sorted_until()`.

**2.53.2.5** `template<typename _ForwardIterator> _ForwardIterator std::is_sorted_until ( _ForwardIterator __first, _ForwardIterator __last )`

Determines the end of a sorted sequence.

**Parameters**

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.

**Returns**

An iterator pointing to the last iterator `i` in `[__first, __last)` for which the range `[__first, i)` is sorted.

Definition at line 3980 of file `stl_algo.h`.

**2.53.2.6** `template<typename _ForwardIterator, typename _Compare> _ForwardIterator std::is_sorted_until ( _ForwardIterator __first, _ForwardIterator __last, _Compare __comp )`

Determines the end of a sorted sequence using comparison functor.

## Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

## Returns

An iterator pointing to the last iterator `i` in `[__first, __last)` for which the range `[__first, i)` is sorted.

Definition at line 4009 of file `stl_algo.h`.

Referenced by `std::is_sorted()`.

2.53.2.7 `template<typename _I1, typename _I2> bool std::lexicographical_compare ( _I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2 ) [inline]`

Performs **dictionary** comparison on ranges.

## Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.

## Returns

A boolean true or false.

*Returns true if the sequence of elements defined by the range `[first1,last1)` is lexicographically less than the sequence of elements defined by the range `[first2,last2)`. Returns false otherwise.* (Quoted from [25.3.8]/1.) If the iterators are all character pointers, then this is an inline call to `memcmp`.

Definition at line 1084 of file `stl_algobase.h`.

2.53.2.8 `template<typename _I1, typename _I2, typename _Compare> bool std::lexicographical_compare ( _I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2, _Compare __comp )`

Performs **dictionary** comparison on ranges.

## Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.
<code>__comp</code>	A <a href="#">comparison functor</a> .

## Returns

A boolean true or false.

The same as the four-parameter `lexicographical_compare`, but uses the `comp` parameter instead of `<`.

Definition at line 1120 of file `stl_algobase.h`.

Referenced by `std::operator<()`.



2.53.2.9 `template<typename _Tp> const _Tp & std::max ( const _Tp & __a, const _Tp & __b ) [inline]`

This does what you think it does.

## Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.

## Returns

The greater of the parameters.

This is the simple classic generic implementation. It will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 216 of file `stl_algobase.h`.

Referenced by `__gnu_parallel::__parallel_nth_element()`, `__gnu_parallel::__parallel_random_shuffle_drs()`, `std::__Dequeue_base< _Tp, _Alloc >::M_initialize_map()`, `std::discard_block_engine< _RandomNumberEngine, __p, __r >::max()`, `std::shuffle_order_engine< _RandomNumberEngine, __k >::max()`, `std::normal_distribution< result_type >::max()`, `std::lognormal_distribution< _RealType >::max()`, `std::gamma_distribution< result_type >::max()`, `std::chi_squared_distribution< _RealType >::max()`, `std::cauchy_distribution< _RealType >::max()`, `std::fisher_f_distribution< _RealType >::max()`, `std::student_t_distribution< _RealType >::max()`, `std::bernoulli_distribution::max()`, `std::geometric_distribution< _IntType >::max()`, `std::negative_binomial_distribution< _IntType >::max()`, `std::poisson_distribution< _IntType >::max()`, `std::exponential_distribution< _RealType >::max()`, `std::weibull_distribution< _RealType >::max()`, `std::extreme_value_distribution< _RealType >::max()`, `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

**2.53.2.10** `template<typename _Tp, typename _Compare> const _Tp &std::max ( const _Tp &__a, const _Tp &__b, _Compare __comp ) [inline]`

This does what you think it does.

## Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.
<code>__comp</code>	A <a href="#">comparison functor</a> .

## Returns

The greater of the parameters.

This will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 260 of file `stl_algobase.h`.

**2.53.2.11** `template<typename _ForwardIterator> _ForwardIterator std::max_element ( _ForwardIterator __first, _ForwardIterator __last )`

Return the maximum element in a range.

## Parameters

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

## Returns

Iterator referencing the first instance of the largest value.

Definition at line 6284 of file `stl_algo.h`.

2.53.2.12 `template<typename _ForwardIterator, typename _Compare> _ForwardIterator std::max_element ( _ForwardIterator  
__first, _ForwardIterator __last, _Compare __comp )`

Return the maximum element in a range using comparison functor.

## Parameters

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

## Returns

Iterator referencing the first instance of the largest value according to `__comp`.

Definition at line 6312 of file `stl_algo.h`.

2.53.2.13 `template<typename _Inputiterator1, typename _Inputiterator2, typename _Outputiterator > _Outputiterator std::merge ( _Inputiterator1 __first1, _Inputiterator1 __last1, _Inputiterator2 __first2, _Inputiterator2 __last2, _Outputiterator __result )`

Merges two sorted ranges.

## Parameters

<code>__first1</code>	An iterator.
<code>__first2</code>	Another iterator.
<code>__last1</code>	Another iterator.
<code>__last2</code>	Another iterator.
<code>__result</code>	An iterator pointing to the end of the merged range.

## Returns

An iterator pointing to the first element *not less than* *val*.

Merges the ranges `[__first1,__last1)` and `[__first2,__last2)` into the sorted range `[__result, __result + (__last1-__first1) + (__last2-__first2))`. Both input ranges must be sorted, and the output range must not overlap with either of the input ranges. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

Definition at line 5526 of file `stl_algo.h`.

2.53.2.14 `template<typename _Inputiterator1, typename _Inputiterator2, typename _Outputiterator, typename _Compare > _Outputiterator std::merge ( _Inputiterator1 __first1, _Inputiterator1 __last1, _Inputiterator2 __first2, _Inputiterator2 __last2, _Outputiterator __result, _Compare __comp )`

Merges two sorted ranges.

## Parameters

<code>__first1</code>	An iterator.
<code>__first2</code>	Another iterator.
<code>__last1</code>	Another iterator.
<code>__last2</code>	Another iterator.
<code>__result</code>	An iterator pointing to the end of the merged range.
<code>__comp</code>	A functor to use for comparisons.

## Returns

An iterator pointing to the first element "not less than" *val*.

Merges the ranges `[__first1,__last1)` and `[__first2,__last2)` into the sorted range `[__result, __result + (__last1-__first1) + (__last2-__first2))`. Both input ranges must be sorted, and the output range must not overlap with either of the input

ranges. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

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

Definition at line 5590 of file `stl_algo.h`.

**2.53.2.15** `template<typename _Tp> const _Tp & std::min ( const _Tp & __a, const _Tp & __b ) [inline]`

This does what you think it does.

#### Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.

#### Returns

The lesser of the parameters.

This is the simple classic generic implementation. It will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 193 of file `stl_algobase.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, `__gnu_parallel::__parallel_sort_qs_divide()`, `__gnu_profile::__report()`, `__gnu_parallel::__search_template()`, `__gnu_parallel::__sequential_random_shuffle()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`, `std::basic_string<_CharT>::compare()`, `std::discard_block_engine<_RandomNumberEngine, __p, __r>::min()`, `std::shuffle_order_engine<_RandomNumberEngine, __k>::min()`, `std::bernoulli_distribution::min()`, `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

**2.53.2.16** `template<typename _Tp, typename _Compare> const _Tp & std::min ( const _Tp & __a, const _Tp & __b, _Compare __comp ) [inline]`

This does what you think it does.

#### Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.
<code>__comp</code>	A <a href="#">comparison functor</a> .

#### Returns

The lesser of the parameters.

This will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 239 of file `stl_algobase.h`.

**2.53.2.17** `template<typename _ForwardIterator> _ForwardIterator std::min_element ( _ForwardIterator __first, _ForwardIterator __last )`

Return the minimum element in a range.

## Parameters

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

## Returns

Iterator referencing the first instance of the smallest value.

Definition at line 6228 of file `stl_algo.h`.

```
2.53.2.18  template<typename _ForwardIterator, typename _Compare> _ForwardIterator std::min_element ( _ForwardIterator
    __first, _ForwardIterator __last, _Compare __comp )
```

Return the minimum element in a range using comparison functor.

## Parameters

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

## Returns

Iterator referencing the first instance of the smallest value according to `__comp`.

Definition at line 6256 of file `stl_algo.h`.

```
2.53.2.19  template<typename _Tp> pair< const _Tp &, const _Tp &> std::minmax ( const _Tp & __a, const _Tp & __b )
    [inline]
```

Determines min and max at once as an ordered pair.

## Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.

## Returns

A pair(`__b`, `__a`) if `__b` is smaller than `__a`, pair(`__a`, `__b`) otherwise.

Definition at line 4039 of file `stl_algo.h`.

```
2.53.2.20  template<typename _Tp, typename _Compare> pair< const _Tp &, const _Tp &> std::minmax ( const _Tp & __a,
    const _Tp & __b, _Compare __comp ) [inline]
```

Determines min and max at once as an ordered pair.

## Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.

<code>__comp</code>	A <a href="#">comparison functor</a> .
---------------------	--

**Returns**

A pair(`__b`, `__a`) if `__b` is smaller than `__a`, pair(`__a`, `__b`) otherwise.

Definition at line 4059 of file `stl_algo.h`.

**2.53.2.21** `template<typename _ForwardIterator > pair<_ForwardIterator, _ForwardIterator> std::minmax_element (`  
`_ForwardIterator __first, _ForwardIterator __last )`

Return a pair of iterators pointing to the minimum and maximum elements in a range.

**Parameters**

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

**Returns**

make\_pair(m, M), where m is the first iterator i in [`__first`, `__last`) such that no other element in the range is smaller, and where M is the last iterator i in [`__first`, `__last`) such that no other element in the range is larger.

Definition at line 4078 of file `stl_algo.h`.

References `std::make_pair()`.

**2.53.2.22** `template<typename _ForwardIterator, typename _Compare > pair<_ForwardIterator, _ForwardIterator>`  
`std::minmax_element ( _ForwardIterator __first, _ForwardIterator __last, _Compare __comp )`

Return a pair of iterators pointing to the minimum and maximum elements in a range.

**Parameters**

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

**Returns**

make\_pair(m, M), where m is the first iterator i in [`__first`, `__last`) such that no other element in the range is smaller, and where M is the last iterator i in [`__first`, `__last`) such that no other element in the range is larger.

Definition at line 4154 of file `stl_algo.h`.

References `std::make_pair()`.

**2.53.2.23** `template<typename _BidirectionalIterator > bool std::next_permutation ( _BidirectionalIterator __first,`  
`_BidirectionalIterator __last )`

Permute range into the next *dictionary* ordering.

**Parameters**

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

## Returns

False if wrapped to first permutation, true otherwise.

Treats all permutations of the range as a set of *dictionary* sorted sequences. Permutes the current sequence into the next one of this set. Returns true if there are more sequences to generate. If the sequence is the largest of the set, the smallest is generated and false returned.

Definition at line 3654 of file stl\_algo.h.

References std::iter\_swap(), and std::reverse().

**2.53.2.24** `template<typename _BidirectionalIterator, typename _Compare> bool std::next_permutation ( _BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp )`

Permute range into the next *dictionary* ordering using comparison functor.

## Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	A comparison functor.

## Returns

False if wrapped to first permutation, true otherwise.

Treats all permutations of the range `[__first, __last)` as a set of *dictionary* sorted sequences ordered by `__comp`. Permutes the current sequence into the next one of this set. Returns true if there are more sequences to generate. If the sequence is the largest of the set, the smallest is generated and false returned.

Definition at line 3711 of file stl\_algo.h.

References std::iter\_swap(), and std::reverse().

**2.53.2.25** `template<typename _RandomAccessIterator> void std::nth_element ( _RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last ) [inline]`

Sort a sequence just enough to find a particular position.

## Parameters

<code>__first</code>	An iterator.
<code>__nth</code>	Another iterator.
<code>__last</code>	Another iterator.

## Returns

Nothing.

Rearranges the elements in the range `[__first, __last)` so that `*__nth` is the same element that would have been in that position had the whole sequence been sorted. The elements either side of `*__nth` are not completely sorted, but for any iterator *i* in the range `[__first, __nth)` and any iterator *j* in the range `[__nth, __last)` it holds that `*i < *j` is false.

Definition at line 5370 of file stl\_algo.h.

References std::\_\_lg().

**2.53.2.26** `template<typename _RandomAccessIterator, typename _Compare> void std::nth_element ( _RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp ) [inline]`

Sort a sequence just enough to find a particular position using a predicate for comparison.



## Parameters

<code>__first</code>	An iterator.
<code>__nth</code>	Another iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

## Returns

Nothing.

Rearranges the elements in the range `[__first,__last)` so that `*__nth` is the same element that would have been in that position had the whole sequence been sorted. The elements either side of `*__nth` are not completely sorted, but for any iterator *i* in the range `[__first,__nth)` and any iterator *j* in the range `[__nth,__last)` it holds that `__comp(*j,*i)` is false.

Definition at line 5409 of file `stl_algo.h`.

References `std::__lg()`.

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

Sort the smallest elements of a sequence.

## Parameters

<code>__first</code>	An iterator.
<code>__middle</code>	Another iterator.
<code>__last</code>	Another iterator.

## Returns

Nothing.

Sorts the smallest `(__middle-__first)` elements in the range `[first,last)` and moves them to the range `[__first,__middle)`. The order of the remaining elements in the range `[__middle,__last)` is undefined. After the sort if *i* and *j* are iterators in the range `[__first,__middle)` such that *i* precedes *j* and *k* is an iterator in the range `[__middle,__last)` then `*j<*i` and `*k<*i` are both false.

Definition at line 5294 of file `stl_algo.h`.

References `std::__heap_select()`, and `std::sort_heap()`.

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

Sort the smallest elements of a sequence using a predicate for comparison.

## Parameters

<code>__first</code>	An iterator.
<code>__middle</code>	Another iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

**Returns**

Nothing.

Sorts the smallest (`__middle-__first`) elements in the range `[__first,__last)` and moves them to the range `[__first,__middle)`. The order of the remaining elements in the range `[__middle,__last)` is undefined. After the sort if *i* and *j* are iterators in the range `[__first,__middle)` such that *i* precedes *j* and *k* is an iterator in the range `[__middle,__last)` then `*__comp(j,*i)` and `*__comp(*k,*i)` are both false.

Definition at line 5333 of file `stl_algo.h`.

References `std::__heap_select()`, and `std::sort_heap()`.

Referenced by `std::__introsort_loop()`.

**2.53.2.29** `template<typename _InputIterator, typename _RandomAccessIterator> _RandomAccessIterator std::partial_sort_copy (`  
`_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last )`

Copy the smallest elements of a sequence.

**Parameters**

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__result_first</code>	A random-access iterator.
<code>__result_last</code>	Another random-access iterator.

**Returns**

An iterator indicating the end of the resulting sequence.

Copies and sorts the smallest *N* values from the range `[__first,__last)` to the range beginning at `__result_first`, where the number of elements to be copied, *N*, is the smaller of `(__last-__first)` and `(__result_last-__result_first)`. After the sort if *i* and *j* are iterators in the range `[__result_first,__result_first+N)` such that *i* precedes *j* then `*j<*i` is false. The value returned is `__result_first+N`.

Definition at line 1974 of file `stl_algo.h`.

References `std::make_heap()`, and `std::sort_heap()`.

**2.53.2.30** `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare> _RandomAccessIterator`  
`std::partial_sort_copy ( _InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first,`  
`_RandomAccessIterator __result_last, _Compare __comp )`

Copy the smallest elements of a sequence using a predicate for comparison.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	Another input iterator.
<code>__result_first</code>	A random-access iterator.
<code>__result_last</code>	Another random-access iterator.
<code>__comp</code>	A comparison functor.

**Returns**

An iterator indicating the end of the resulting sequence.

Copies and sorts the smallest *N* values from the range `[__first,__last)` to the range beginning at `result_first`, where the number of elements to be copied, *N*, is the smaller of `(__last-__first)` and `(__result_last-__result_first)`. After

the sort if  $i$  and  $j$  are iterators in the range  $[\_\text{result\_first}, \_\text{result\_first}+N)$  such that  $i$  precedes  $j$  then  $\_\_\text{comp}(*j, *i)$  is false. The value returned is  $\_\text{result\_first}+N$ .

Definition at line 2040 of file `stl_algo.h`.

References `std::make_heap()`, and `std::sort_heap()`.

**2.53.2.31** `template<typename _BidirectionalIterator > bool std::prev_permutation ( _BidirectionalIterator __first, _BidirectionalIterator __last )`

Permute range into the previous *dictionary* ordering.

Parameters

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

Returns

False if wrapped to last permutation, true otherwise.

Treats all permutations of the range as a set of *dictionary* sorted sequences. Permutes the current sequence into the previous one of this set. Returns true if there are more sequences to generate. If the sequence is the smallest of the set, the largest is generated and false returned.

Definition at line 3767 of file `stl_algo.h`.

References `std::iter_swap()`, and `std::reverse()`.

**2.53.2.32** `template<typename _BidirectionalIterator, typename _Compare > bool std::prev_permutation ( _BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp )`

Permute range into the previous *dictionary* ordering using comparison functor.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	A comparison functor.

Returns

False if wrapped to last permutation, true otherwise.

Treats all permutations of the range  $[\_\text{first}, \_\text{last})$  as a set of *dictionary* sorted sequences ordered by  $\_\_\text{comp}$ . Permutes the current sequence into the previous one of this set. Returns true if there are more sequences to generate. If the sequence is the smallest of the set, the largest is generated and false returned.

Definition at line 3824 of file `stl_algo.h`.

References `std::iter_swap()`, and `std::reverse()`.

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

Sort the elements of a sequence.

## Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.

## Returns

Nothing.

Sorts the elements in the range `[__first,__last)` in ascending order, such that for each iterator  $i$  in the range `[__first,__last-1)`,  $*(i+1) < *i$  is false.

The relative ordering of equivalent elements is not preserved, use `stable_sort()` if this is needed.

Definition at line 5447 of file `stl_algo.h`.

References `std::__final_insertion_sort()`, `std::__introsort_loop()`, and `std::__lg()`.

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

Sort the elements of a sequence using a predicate for comparison.

## Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

## Returns

Nothing.

Sorts the elements in the range `[__first,__last)` in ascending order, such that `__comp(*(i+1),*i)` is false for every iterator  $i$  in the range `[__first,__last-1)`.

The relative ordering of equivalent elements is not preserved, use `stable_sort()` if this is needed.

Definition at line 5483 of file `stl_algo.h`.

References `std::__final_insertion_sort()`, `std::__introsort_loop()`, and `std::__lg()`.

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

Sort the elements of a sequence, preserving the relative order of equivalent elements.

## Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.

## Returns

Nothing.

Sorts the elements in the range `[__first,__last)` in ascending order, such that for each iterator  $i$  in the range `[__first,__last-1)`,  $*(i+1) < *i$  is false.

The relative ordering of equivalent elements is preserved, so any two elements  $x$  and  $y$  in the range `[__first,__last)` such that  $x < y$  is false and  $y < x$  is false will have the same relative ordering after calling `stable_sort()`.

Definition at line 5649 of file `stl_algo.h`.

References `std::__inplace_stable_sort()`, `std::Temporary_buffer<_ForwardIterator, _Tp>::begin()`, and `std::__Temporary_buffer<_ForwardIterator, _Tp>::size()`.

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

Sort the elements of a sequence using a predicate for comparison, preserving the relative order of equivalent elements.

#### Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

#### Returns

Nothing.

Sorts the elements in the range `[__first, __last)` in ascending order, such that for each iterator `i` in the range `[__first, __last-1)`, `__comp(*(i+1), *i)` is false.

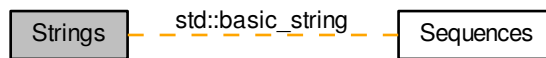
The relative ordering of equivalent elements is preserved, so any two elements `x` and `y` in the range `[__first, __last)` such that `__comp(x, y)` is false and `__comp(y, x)` is false will have the same relative ordering after calling `stable_sort()`.

Definition at line 5691 of file `stl_algo.h`.

References `std::__inplace_stable_sort()`, `std::Temporary_buffer<_ForwardIterator, _Tp>::begin()`, and `std::__Temporary_buffer<_ForwardIterator, _Tp>::size()`.

## 2.54 Strings

Collaboration diagram for Strings:



### Classes

- class [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >](#)
- struct [std::char\\_traits< \\_CharT >](#)

### Typedefs

- typedef [basic\\_string< char >](#) [std::string](#)
- typedef [basic\\_string< char16\\_t >](#) [std::u16string](#)
- typedef [basic\\_string< char32\\_t >](#) [std::u32string](#)
- typedef [basic\\_string< wchar\\_t >](#) [std::wstring](#)

#### 2.54.1 Detailed Description

#### 2.54.2 Typedef Documentation

##### 2.54.2.1 typedef [basic\\_string<char>](#) [std::string](#)

A string of `char`.

Definition at line 62 of file `stringfwd.h`.

##### 2.54.2.2 typedef [basic\\_string<char16\\_t>](#) [std::u16string](#)

A string of `char16_t`.

Definition at line 78 of file `stringfwd.h`.

##### 2.54.2.3 typedef [basic\\_string<char32\\_t>](#) [std::u32string](#)

A string of `char32_t`.

Definition at line 81 of file `stringfwd.h`.

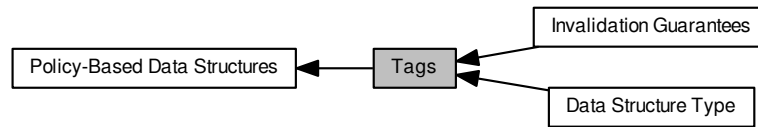
##### 2.54.2.4 typedef [basic\\_string<wchar\\_t>](#) [std::wstring](#)

A string of `wchar_t`.

Definition at line 68 of file `stringfwd.h`.

## 2.55 Tags

Collaboration diagram for Tags:



### Modules

- [Data Structure Type](#)
- [Invalidation Guarantees](#)

### Classes

- [struct `\_\_gnu\_pbds::trivial\_iterator\_tag`](#)

### Typedefs

- [typedef void `\_\_gnu\_pbds::trivial\_iterator\_difference\_type`](#)

#### 2.55.1 Detailed Description

#### 2.55.2 Typedef Documentation

##### 2.55.2.1 typedef void `__gnu_pbds::trivial_iterator_difference_type`

Prohibit moving trivial iterators.

Definition at line 79 of file `tag_and_trait.hpp`.

## 2.56 Traits

Collaboration diagram for Traits:



### Classes

- struct `__gnu_pbds::container_traits< Cntnr >`
- struct `__gnu_pbds::container_traits_base< _Tag >`
- struct `__gnu_pbds::container_traits_base< binary_heap_tag >`
- struct `__gnu_pbds::container_traits_base< binomial_heap_tag >`
- struct `__gnu_pbds::container_traits_base< cc_hash_tag >`
- struct `__gnu_pbds::container_traits_base< gp_hash_tag >`
- struct `__gnu_pbds::container_traits_base< list_update_tag >`
- struct `__gnu_pbds::container_traits_base< ov_tree_tag >`
- struct `__gnu_pbds::container_traits_base< pairing_heap_tag >`
- struct `__gnu_pbds::container_traits_base< pat_trie_tag >`
- struct `__gnu_pbds::container_traits_base< rb_tree_tag >`
- struct `__gnu_pbds::container_traits_base< rc_binomial_heap_tag >`
- struct `__gnu_pbds::container_traits_base< splay_tree_tag >`
- struct `__gnu_pbds::container_traits_base< thin_heap_tag >`
- struct `__gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >`
- struct `__gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >`
- struct `__gnu_pbds::detail::no_throw_copies< Key, Mapped >`
- struct `__gnu_pbds::detail::no_throw_copies< Key, null_type >`
- struct `__gnu_pbds::detail::stored_data< _Tv, _Th >`
- struct `__gnu_pbds::detail::stored_data< _Tv, null_type >`
- struct `__gnu_pbds::detail::stored_hash< _Th >`
- struct `__gnu_pbds::detail::stored_value< _Tv >`
- struct `__gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp >`
- struct `__gnu_pbds::detail::tree_metadata_helper< Node_Update, false >`
- struct `__gnu_pbds::detail::tree_metadata_helper< Node_Update, true >`
- struct `__gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp >`
- struct `__gnu_pbds::detail::trie_metadata_helper< Node_Update, false >`
- struct `__gnu_pbds::detail::trie_metadata_helper< Node_Update, true >`



- struct `__gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >`
- struct `__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >`
- struct `__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >`
- struct `__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, Store_Hash >`
- struct `__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, false >`
- struct `__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, true >`
- struct `__gnu_pbds::detail::type_base< Key, null_type, _Alloc, false >`
- struct `__gnu_pbds::detail::type_base< Key, null_type, _Alloc, true >`
- struct `__gnu_pbds::detail::type_dispatch< Key, Mapped, _Alloc, Store_Hash >`
- struct `__gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >`
- struct `__gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >`
- struct `__gnu_pbds::null_type`

## Enumerations

- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- enum { `__gnu_pbds::container_traits< Cntnr >::order_preserving`, `__gnu_pbds::container_traits< Cntnr >::erase_can_throw`, `__gnu_pbds::container_traits< Cntnr >::split_join_can_throw`, `__gnu_pbds::container_traits< Cntnr >::reverse_iteration` }
- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }

## Variables

- static null\_type `__gnu_pbds::detail::type_base< Key, null_type, _Alloc, false >::s_null_type`
- static null\_type `__gnu_pbds::detail::type_base< Key, null_type, _Alloc, true >::s_null_type`

### 2.56.1 Detailed Description

### 2.56.2 Enumeration Type Documentation

#### 2.56.2.1 `template<typename Cntnr > anonymous enum`

## Enumerator

***order\_preserving*** True only if Cntnr objects guarantee storing keys by order.

***erase\_can\_throw*** True only if erasing a key can throw.

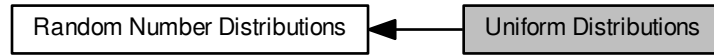
***split\_join\_can\_throw*** True only if split or join operations can throw.

***reverse\_iteration*** True only reverse iterators are supported.

Definition at line 426 of file `tag_and_trait.hpp`.

## 2.57 Uniform Distributions

Collaboration diagram for Uniform Distributions:



### Classes

- class `std::uniform_int_distribution<_IntType>`
- struct `std::uniform_int_distribution<_IntType>::param_type`
- class `std::uniform_real_distribution<_RealType>`
- struct `std::uniform_real_distribution<_RealType>::param_type`

### Functions

- template<typename \_IntType> bool `std::operator!=` (const `std::uniform_int_distribution<_IntType>` &\_\_d1, const `std::uniform_int_distribution<_IntType>` &\_\_d2)
- template<typename \_IntType> bool `std::operator!=` (const `std::uniform_real_distribution<_IntType>` &\_\_d1, const `std::uniform_real_distribution<_IntType>` &\_\_d2)
- template<typename \_IntType, typename \_CharT, typename \_Traits> `std::basic_ostream<_CharT, _Traits>` & `std::operator<<` (`std::basic_ostream<_CharT, _Traits>` &, const `std::uniform_int_distribution<_IntType>` &)
- template<typename \_RealType, typename \_CharT, typename \_Traits> `std::basic_ostream<_CharT, _Traits>` & `std::operator<<` (`std::basic_ostream<_CharT, _Traits>` &, const `std::uniform_real_distribution<_RealType>` &)
- template<typename \_IntType, typename \_CharT, typename \_Traits> `std::basic_istream<_CharT, _Traits>` & `std::operator>>` (`std::basic_istream<_CharT, _Traits>` &, `std::uniform_int_distribution<_IntType>` &)
- template<typename \_RealType, typename \_CharT, typename \_Traits> `std::basic_istream<_CharT, _Traits>` & `std::operator>>` (`std::basic_istream<_CharT, _Traits>` &, `std::uniform_real_distribution<_RealType>` &)

#### 2.57.1 Detailed Description

#### 2.57.2 Function Documentation

**2.57.2.1** template<typename \_IntType> bool `std::operator!=` ( const `std::uniform_int_distribution<_IntType>` & \_\_d1, const `std::uniform_int_distribution<_IntType>` & \_\_d2 ) [inline]

Return true if two uniform integer distributions have different parameters.

Definition at line 1825 of file random.h.

**2.57.2.2** template<typename \_IntType> bool `std::operator!=` ( const `std::uniform_real_distribution<_IntType>` & \_\_d1, const `std::uniform_real_distribution<_IntType>` & \_\_d2 ) [inline]

Return true if two uniform real distributions have different parameters.

Definition at line 2034 of file random.h.

2.57.2.3 `template<typename _IntType , typename _CharT , typename _Traits > std::basic_ostream<_CharT, _Traits>&  
std::operator<< ( std::basic_ostream<_CharT, _Traits > & , const std::uniform_int_distribution<_IntType > & )`

Inserts a `uniform_int_distribution` random number distribution `___x` into the output stream `os`.

## Parameters

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

## Returns

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

**2.57.2.4** `template<typename _RealType , typename _CharT , typename _Traits > std::basic_ostream<_CharT, _Traits>& std::operator<< ( std::basic_ostream<_CharT, _Traits> & , const std::uniform_real_distribution<_RealType> & )`

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

## Parameters

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

## Returns

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

**2.57.2.5** `template<typename _IntType , typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>& std::operator>> ( std::basic_istream<_CharT, _Traits> & , std::uniform_int_distribution<_IntType> & )`

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

## Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>uniform_int_distribution</code> random number generator engine.

## Returns

The input stream with `__x` extracted or in an error state.

**2.57.2.6** `template<typename _RealType , typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>& std::operator>> ( std::basic_istream<_CharT, _Traits> & , std::uniform_real_distribution<_RealType> & )`

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

## Parameters

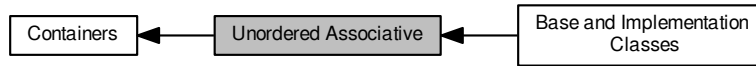
<code>__is</code>	An input stream.
<code>__x</code>	A <code>uniform_real_distribution</code> random number generator engine.

## Returns

The input stream with `__x` extracted or in an error state.

## 2.58 Unordered Associative

Collaboration diagram for Unordered Associative:



### Modules

- [Base and Implementation Classes](#)

### Classes

- class `std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>`
- class `std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>`
- class `std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>`
- class `std::unordered_set<_Value, _Hash, _Pred, _Alloc>`

### 2.58.1 Detailed Description

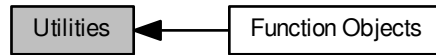
Unordered associative containers allow fast retrieval of data based on keys.

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

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

## 2.59 Utilities

Collaboration diagram for Utilities:



### Modules

- [Function Objects](#)

### Classes

- [struct `std::pair<\_T1, \_T2>`](#)
- [struct `std::piecewise\_construct\_t`](#)

### Functions

- `template<typename _Tp> _Tp * std::\_\_addressof (_Tp &__r) noexcept`
- `template<typename _Tp> _Tp * std::addressof (_Tp &__r) noexcept`
- `template<typename _Tp> constexpr _Tp && std::forward (typename std::remove_reference<_Tp>::type &__t) noexcept`
- `template<typename _Tp> constexpr _Tp && std::forward (typename std::remove_reference<_Tp>::type &&__t) noexcept`
- `template<class _T1, class _T2> constexpr pair< typename __decay_and_strip<_T1>::__type, typename __decay_and_strip<_T2>::__type> std::make\_pair (_T1 &&__x, _T2 &&__y)`
- `template<typename _Tp> constexpr std::remove_reference<_Tp>::type && std::move (_Tp &&__t) noexcept`
- `template<typename _Tp> constexpr conditional< __move_if_noexcept_cond<_Tp>::value, const _Tp &, _Tp &&>::type std::move\_if\_noexcept (_Tp &__x) noexcept`
- `template<class _T1, class _T2> constexpr bool std::operator!= (const pair<_T1, _T2> &__x, const pair<_T1, _T2> &__y)`
- `template<class _T1, class _T2> constexpr bool std::operator< (const pair<_T1, _T2> &__x, const pair<_T1, _T2> &__y)`
- `template<class _T1, class _T2> constexpr bool std::operator<= (const pair<_T1, _T2> &__x, const pair<_T1, _T2> &__y)`
- `template<class _T1, class _T2> constexpr bool std::operator== (const pair<_T1, _T2> &__x, const pair<_T1, _T2> &__y)`
- `template<class _T1, class _T2> constexpr bool std::operator> (const pair<_T1, _T2> &__x, const pair<_T1, _T2> &__y)`
- `template<class _T1, class _T2> constexpr bool std::operator>= (const pair<_T1, _T2> &__x, const pair<_T1, _T2> &__y)`
- `template<class _T1, class _T2> void std::swap (pair<_T1, _T2> &__x, pair<_T1, _T2> &__y) noexcept(noexcept(__x.swap(__y)))`

- `template<typename _Tp > void std::swap ( _Tp &__a, _Tp &__b) noexcept(__and_< is_nothrow_move_constructible< _Tp >,is_nothrow_move_assignable< _Tp >>::value)`
- `template<typename _Tp, size_t _Nm> void std::swap ( _Tp(&__a)[ _Nm], _Tp(&__b)[ _Nm]) noexcept(noexcept(swap(*__a,*__b)))`

## Variables

- `constexpr piecewise_construct_t std::piecewise_construct`

### 2.59.1 Detailed Description

### 2.59.2 Function Documentation

#### 2.59.2.1 `template<typename _Tp > _Tp* std::__addressof ( _Tp &__r ) [inline],[noexcept]`

Same as C++11 `std::addressof`.

Definition at line 47 of file `move.h`.

Referenced by `std::_Destroy()`, `std::addressof()`, and `std::vector< _State >::data()`.

#### 2.59.2.2 `template<typename _Tp > _Tp* std::addressof ( _Tp &__r ) [inline],[noexcept]`

Returns the actual address of the object or function referenced by `r`, even in the presence of an overloaded operator`&`.

#### Parameters

<code>__r</code>	Reference to an object or function.
------------------	-------------------------------------

#### Returns

The actual address.

Definition at line 135 of file `move.h`.

References `std::__addressof()`.

Referenced by `std::pointer_traits< _Tp * >::pointer_to()`.

#### 2.59.2.3 `template<typename _Tp > constexpr _Tp&& std::forward ( typename std::remove_reference< _Tp >::type & __t ) [noexcept]`

Forward an lvalue.

#### Returns

The parameter cast to the specified type.

This function is used to implement "perfect forwarding".

Definition at line 76 of file `move.h`.

#### 2.59.2.4 `template<typename _Tp > constexpr _Tp&& std::forward ( typename std::remove_reference< _Tp >::type && __t ) [noexcept]`

Forward an rvalue.

**Returns**

The parameter cast to the specified type.

This function is used to implement "perfect forwarding".

Definition at line 87 of file move.h.

**2.59.2.5** `template<class _T1 , class _T2 > constexpr pair<typename __decay_and_strip<_T1>::__type, typename __decay_and_strip<_T2>::__type> std::make_pair ( _T1 && __x, _T2 && __y )`

A convenience wrapper for creating a pair from two objects.

**Parameters**

<code>__x</code>	The first object.
<code>__y</code>	The second object.

**Returns**

A newly-constructed pair<> object of the appropriate type.

The standard requires that the objects be passed by reference-to-const, but LWG issue #181 says they should be passed by const value. We follow the LWG by default.

Definition at line 276 of file stl\_pair.h.

Referenced by `__gnu_parallel::__find_template()`, `__gnu_debug::__get_distance()`, `__gnu_parallel::__parallel_merge()`, `__gnu_parallel::__parallel_sort_qsb()`, `__gnu_parallel::__qsb_local_sort_with_helping()`, `__gnu_parallel::__find_if_selector::M_sequential_algorithm()`, `__gnu_parallel::__adjacent_find_selector::M_sequential_algorithm()`, `__gnu_parallel::__find_first_of_selector<_FIterator>::M_sequential_algorithm()`, `std::minmax_element()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, `__gnu_parallel::parallel_multiway_merge()`, `__gnu_parallel::parallel_sort_mwms_pu()`, and `__gnu_pbds::detail::pat_trie_base::Node_citer< Node, Leaf, Head, Inode, C_iterator, Iterator, _Alloc >::valid_prefix()`.

**2.59.2.6** `template<typename _Tp > constexpr std::remove_reference<_Tp>::__type&& std::move ( _Tp && __t ) [noexcept]`

Convert a value to an rvalue.

**Parameters**

<code>__t</code>	A thing of arbitrary type.
------------------	----------------------------

**Returns**

The parameter cast to an rvalue-reference to allow moving it.

Definition at line 101 of file move.h.

Referenced by `std::basic_regex<_Ch_type, _Rx_traits>::assign()`, `std::unordered_set<_Value, _Hash, _Pred, _Alloc>::insert()`, `std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::insert()`, `std::vector<_State>::insert()`, `std::list<__inp, __rebind_inp>::insert()`, `std::deque<_Tp, _Alloc>::insert()`, `std::forward_list<_Tp, _Alloc>::merge()`, `std::move_if_noexcept()`, `std::vector<_State>::operator=()`, `std::basic_regex<_Ch_type, _Rx_traits>::operator=()`, `std::forward_list<_Tp, _Alloc>::operator=()`, `std::match_results<_FwdIterT, _Alloc>::operator=()`, and `std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::operator[]()`.

**2.59.2.7** `template<typename _Tp > constexpr conditional<__move_if_noexcept_cond<_Tp>::value, const _Tp&, _Tp&&>::type std::move_if_noexcept ( _Tp & __x ) [inline], [noexcept]`

Conditionally convert a value to an rvalue.



## Parameters

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

## Returns

The parameter, possibly cast to an rvalue-reference.

Same as `std::move` unless the type's move constructor could throw and the type is copyable, in which case an lvalue-reference is returned instead.

Definition at line 121 of file `move.h`.

References `std::move()`.

**2.59.2.8** `template<class _T1, class _T2 > constexpr bool std::operator!= ( const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y ) [inline]`

Uses `operator==` to find the result.

Definition at line 227 of file `stl_pair.h`.

**2.59.2.9** `template<class _T1, class _T2 > constexpr bool std::operator< ( const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y ) [inline]`

<http://gcc.gnu.org/onlinedocs/libstdc++/manual/utilities.html>

Definition at line 220 of file `stl_pair.h`.

References `std::pair< _T1, _T2 >::first`.

**2.59.2.10** `template<class _T1, class _T2 > constexpr bool std::operator<= ( const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y ) [inline]`

Uses `operator<` to find the result.

Definition at line 239 of file `stl_pair.h`.

**2.59.2.11** `template<class _T1, class _T2 > constexpr bool std::operator== ( const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y ) [inline]`

Two pairs of the same type are equal iff their members are equal.

Definition at line 214 of file `stl_pair.h`.

References `std::pair< _T1, _T2 >::first`, and `std::pair< _T1, _T2 >::second`.

**2.59.2.12** `template<class _T1, class _T2 > constexpr bool std::operator> ( const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y ) [inline]`

Uses `operator<` to find the result.

Definition at line 233 of file `stl_pair.h`.

**2.59.2.13** `template<class _T1, class _T2 > constexpr bool std::operator>= ( const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y ) [inline]`

Uses `operator<` to find the result.

Definition at line 245 of file `stl_pair.h`.

**2.59.2.14** `template<class _T1, class _T2> void std::swap ( pair<_T1, _T2> &__x, pair<_T1, _T2> &__y ) [inline], [noexcept]`

See `std::pair::swap()`.

Definition at line 254 of file `stl_pair.h`.

**2.59.2.15** `template<typename _Tp> void std::swap ( _Tp &__a, _Tp &__b ) const [inline], [noexcept]`

Swaps two values.

Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.

Returns

Nothing.

Definition at line 166 of file `move.h`.

Referenced by `__gnu_parallel::LoserTree< __stable, _Tp, _Compare >::__delete_min_insert()`, `__gnu_parallel::LoserTree< false, _Tp, _Compare >::__delete_min_insert()`, `std::__rotate()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::__Safe_iterator()`, `std::regex_traits< _Ch_type >::imbue()`, `std::iter_swap()`, `std::swap()`, `std::basic_regex< _Ch_type, _Rx_traits >::swap()`, `std::forward_list< _Tp, _Alloc >::swap()`, and `std::deque< _Tp, _Alloc >::swap()`.

**2.59.2.16** `template<typename _Tp, size_t _Nm> void std::swap ( _Tp(&) __a[_Nm], _Tp(&) __b[_Nm] ) [inline], [noexcept]`

Swap the contents of two arrays.

Definition at line 185 of file `move.h`.

References `std::swap()`.

## 2.59.3 Variable Documentation

**2.59.3.1** `constexpr piecewise_construct_t std::piecewise_construct`

`piecewise_construct`

Definition at line 79 of file `stl_pair.h`.

Referenced by `std::map< _Key, _Tp, _Compare, _Alloc >::operator[]()`.

## 3 Namespace Documentation

### 3.1 `__gnu_cxx` Namespace Reference

#### Namespaces

- [\\_\\_detail](#)
- [typelist](#)

#### Classes

- [struct \\_\\_alloc\\_traits](#)
- [struct \\_\\_common\\_pool\\_policy](#)
- [class \\_\\_mt\\_alloc](#)
- [class \\_\\_mt\\_alloc\\_base](#)
- [struct \\_\\_per\\_type\\_pool\\_policy](#)
- [class \\_\\_pool](#)
- [class \\_\\_pool< false >](#)
- [class \\_\\_pool< true >](#)
- [class \\_\\_pool\\_alloc](#)
- [class \\_\\_pool\\_alloc\\_base](#)
- [struct \\_\\_pool\\_base](#)
- [class \\_\\_rc\\_string\\_base](#)
- [class \\_\\_scoped\\_lock](#)
- [class \\_\\_versa\\_string](#)
- [struct \\_Caster](#)
- [struct \\_Char\\_types](#)
- [class \\_ExtPtr\\_allocator](#)
- [struct \\_Invalid\\_type](#)
- [class \\_Pointer\\_adapter](#)
- [class \\_Relative\\_pointer\\_impl](#)
- [class \\_Relative\\_pointer\\_impl< const \\_Tp >](#)
- [class \\_Std\\_pointer\\_impl](#)
- [struct \\_Unqualified\\_type](#)
- [struct annotate\\_base](#)
- [class array\\_allocator](#)
- [class array\\_allocator\\_base](#)
- [class bitmap\\_allocator](#)
- [struct char\\_traits](#)
- [struct character](#)
- [struct condition\\_base](#)
- [class debug\\_allocator](#)
- [class enc\\_filebuf](#)
- [struct encoding\\_char\\_traits](#)
- [class encoding\\_state](#)
- [struct forced\\_error](#)
- [class free\\_list](#)
- [struct limit\\_condition](#)
- [class malloc\\_allocator](#)
- [class new\\_allocator](#)

- struct [random\\_condition](#)
- class [recursive\\_init\\_error](#)
- class [stdio\\_filebuf](#)
- class [stdio\\_sync\\_filebuf](#)
- class [throw\\_allocator\\_base](#)
- struct [throw\\_allocator\\_limit](#)
- struct [throw\\_allocator\\_random](#)
- struct [throw\\_value\\_base](#)
- struct [throw\\_value\\_limit](#)
- struct [throw\\_value\\_random](#)

### Typedefs

- typedef void(\* [\\_\\_destroy\\_handler](#)) (void \*)
- typedef [\\_\\_versa\\_string](#)< char, [std::char\\_traits](#)< char >, [std::allocator](#)< char >, [\\_\\_rc\\_string\\_base](#) > [\\_\\_rc\\_string](#)
- typedef [\\_\\_vstring](#) [\\_\\_sso\\_string](#)
- typedef [\\_\\_versa\\_string](#)< char16\_t, [std::char\\_traits](#)< char16\_t >, [std::allocator](#)< char16\_t >, [\\_\\_rc\\_string\\_base](#) > [\\_\\_u16rc\\_string](#)
- typedef [\\_\\_u16vstring](#) [\\_\\_u16sso\\_string](#)
- typedef [\\_\\_versa\\_string](#)< char16\_t > [\\_\\_u16vstring](#)
- typedef [\\_\\_versa\\_string](#)< char32\_t, [std::char\\_traits](#)< char32\_t >, [std::allocator](#)< char32\_t >, [\\_\\_rc\\_string\\_base](#) > [\\_\\_u32rc\\_string](#)
- typedef [\\_\\_u32vstring](#) [\\_\\_u32sso\\_string](#)
- typedef [\\_\\_versa\\_string](#)< char32\_t > [\\_\\_u32vstring](#)
- typedef [\\_\\_versa\\_string](#)< char > [\\_\\_vstring](#)
- typedef [\\_\\_versa\\_string](#)< wchar\_t, [std::char\\_traits](#)< wchar\_t >, [std::allocator](#)< wchar\_t >, [\\_\\_rc\\_string\\_base](#) > [\\_\\_wrc\\_string](#)
- typedef [\\_\\_wvstring](#) [\\_\\_wsso\\_string](#)
- typedef [\\_\\_versa\\_string](#)< wchar\_t > [\\_\\_wvstring](#)

### Enumerations

- enum { [\\_S\\_num\\_primes](#) }
- enum [\\_Lock\\_policy](#) { [\\_S\\_single](#), [\\_S\\_mutex](#), [\\_S\\_atomic](#) }

### Functions

- static void [\\_\\_atomic\\_add\\_single](#) (\_Atomic\_word \* \_\_mem, int \_\_val)
- else [\\_\\_atomic\\_add\\_single](#) (\_\_mem, \_\_val)
- [\\_Atomic\\_word](#) [\\_\\_attribute\\_\\_\(\(\\_\\_unused\\_\\_\)\)](#) [\\_\\_exchange\\_and\\_add](#)(volatile \_Atomic\_word \*
- template<class \_Tp > void [\\_\\_aux\\_require\\_boolean\\_expr](#) (const \_Tp & \_\_t)
- template<typename \_ToType, typename \_FromType > \_ToType [\\_\\_const\\_pointer\\_cast](#) (const \_FromType & \_\_arg)
- template<typename \_ToType, typename \_FromType > \_ToType [\\_\\_const\\_pointer\\_cast](#) (\_FromType \* \_\_arg)
- template<typename \_ToType, typename \_FromType > \_ToType [\\_\\_dynamic\\_pointer\\_cast](#) (const \_FromType & \_\_arg)
- template<typename \_ToType, typename \_FromType > \_ToType [\\_\\_dynamic\\_pointer\\_cast](#) (\_FromType \* \_\_arg)
- void [\\_\\_error\\_type\\_must\\_be\\_a\\_signed\\_integer\\_type](#) ()
- void [\\_\\_error\\_type\\_must\\_be\\_an\\_integer\\_type](#) ()
- void [\\_\\_error\\_type\\_must\\_be\\_an\\_unsigned\\_integer\\_type](#) ()
- static \_Atomic\_word [\\_\\_exchange\\_and\\_add\\_single](#) (\_Atomic\_word \* \_\_mem, int \_\_val)
- else return [\\_\\_exchange\\_and\\_add\\_single](#) (\_\_mem, \_\_val)

- `template<class _Concept> void __function_requires ()`
- `template<typename _Type> bool __is_null_pointer (_Type * __ptr)`
- `template<typename _Type> bool __is_null_pointer (_Type)`
- `template<typename _ToType, typename _FromType> _ToType __reinterpret_pointer_cast (const _FromType & __arg)`
- `template<typename _ToType, typename _FromType> _ToType __reinterpret_pointer_cast (_FromType * __arg)`
- `template<typename _ToType, typename _FromType> _ToType __static_pointer_cast (const _FromType & __arg)`
- `template<typename _ToType, typename _FromType> _ToType __static_pointer_cast (_FromType * __arg)`
- `size_t __stl_hash_string (const char * __s)`
- `unsigned long __stl_next_prime (unsigned long __n)`
- `template<typename _TRet, typename _Ret = _TRet, typename _CharT, typename... _Base> _Ret __stoa (_TRet(* __convf)(const _CharT *, _CharT **, _Base...), const char * __name, const _CharT * __str, std::size_t * __idx, _Base... __base)`
- `void __throw_concurrency_lock_error ()`
- `void __throw_concurrency_unlock_error ()`
- `void __throw_forced_error ()`
- `template<typename _String, typename _CharT = typename _String::value_type> _String __to_xstring (int(* __convf)(_CharT *, std::size_t, const _CharT *, __builtin_va_list), std::size_t __n, const _CharT * __fmt,...)`
- `size_t __Bit_scan_forward (size_t __num)`
- `template<class _CharT, class _Traits> void __Rope_fill (basic_ostream< _CharT, _Traits> & __o, size_t __n)`
- `template<class _CharT> bool __Rope_is_simple (_CharT *)`
- `bool __Rope_is_simple (char *)`
- `bool __Rope_is_simple (wchar_t *)`
- `template<class _Rope_iterator> void __Rope_rotate (_Rope_iterator __first, _Rope_iterator __middle, _Rope_iterator __last)`
- `static _Atomic_word int __val if (__pthread_active_p()) return __exchange_and_add(__mem`
- `template<typename _Tp> bool operator!= (const new_allocator< _Tp> &, const new_allocator< _Tp> &)`
- `template<typename _Tp> bool operator!= (const malloc_allocator< _Tp> &, const malloc_allocator< _Tp> &)`
- `template<typename _Tp, typename _Array> bool operator!= (const array_allocator< _Tp, _Array> &, const array_allocator< _Tp, _Array> &)`
- `template<typename _Tp> bool operator!= (const __pool_alloc< _Tp> &, const __pool_alloc< _Tp> &)`
- `template<typename _Tp1, typename _Tp2> bool operator!= (const _Pointer_adapter< _Tp1> & __lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2> bool operator!= (_Tp1 __lhs, const _Pointer_adapter< _Tp2> & __rhs)`
- `template<typename _Tp1, typename _Tp2> bool operator!= (const _Pointer_adapter< _Tp1> & __lhs, const _Pointer_adapter< _Tp2> & __rhs)`
- `template<typename _Tp> bool operator!= (const _Pointer_adapter< _Tp> & __lhs, int __rhs)`
- `template<typename _Tp> bool operator!= (int __lhs, const _Pointer_adapter< _Tp> & __rhs)`
- `template<typename _Tp> bool operator!= (const _Pointer_adapter< _Tp> & __lhs, const _Pointer_adapter< _Tp> & __rhs)`
- `template<typename _Tp, typename _Cond> bool operator!= (const throw_allocator_base< _Tp, _Cond> &, const throw_allocator_base< _Tp, _Cond> &)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All> bool operator!= (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All> & __ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All> & __ht2)`
- `template<typename _Tp, typename _Poolp> bool operator!= (const __mt_alloc< _Tp, _Poolp> &, const __mt_alloc< _Tp, _Poolp> &)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container> bool operator!= (const __normal_iterator< _IteratorL, _Container> & __lhs, const __normal_iterator< _IteratorR, _Container> & __rhs)`
- `template<typename _Iterator, typename _Container> bool operator!= (const __normal_iterator< _Iterator, _Container> & __lhs, const __normal_iterator< _Iterator, _Container> & __rhs)`
- `template<typename _Tp1, typename _Tp2> bool operator!= (const bitmap_allocator< _Tp1> &, const bitmap_allocator< _Tp2> &) throw ()`

- [illegible]

- `template<typename _IteratorL, typename _IteratorR, typename _Container> auto operator- (const __normal_iterator< _↵  
_IteratorL, _Container> &__lhs, const __normal_iterator< _IteratorR, _Container> &__rhs) -> decltype(__↵  
__lhs.base()-__rhs.base())`
- `template<typename _Iterator, typename _Container> __normal_iterator< _Iterator, _Container>::difference_type  
operator- (const __normal_iterator< _Iterator, _Container> &__lhs, const __normal_iterator< _Iterator, ↵  
_Container> &__rhs)`
- `template<typename V, typename I, typename S> bool operator< (const character< V, I, S> &lhs, const character< V,  
I, S> &rhs)`
- `template<typename _Tp1, typename _Tp2> bool operator< (_Tp1 __lhs, const _Pointer_adapter< _Tp2> &__rhs)`
- `template<typename _Tp1, typename _Tp2> bool operator< (const _Pointer_adapter< _Tp1> &__lhs, const ↵  
_Pointer_adapter< _Tp2> &__rhs)`
- `template<typename _Tp1, typename _Tp2> bool operator< (const _Pointer_adapter< _Tp1> &__lhs, _Tp2 __rhs)`
- `template<typename _Cond> bool operator< (const throw_value_base< _Cond> &__a, const throw_value_base<  
_Cond> &__b)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container> bool operator< (const __normal_iterator< ↵  
_IteratorL, _Container> &__lhs, const __normal_iterator< _IteratorR, _Container> &__rhs)`
- `template<typename _Iterator, typename _Container> bool operator< (const __normal_iterator< _Iterator, _Container>  
&__lhs, const __normal_iterator< _Iterator, _Container> &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename> class _Base> bool  
operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base> &__lhs, const __versa_string< _CharT, ↵  
_Traits, _Alloc, _Base> &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename> class _Base> bool  
operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base> &__lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename> class _Base> bool  
operator< (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base> &__rhs)`
- `std::ostream & operator<< (std::ostream &os, const annotate_base &__b)`
- `template<typename _CharT, typename _Traits, typename _StoreT> std::basic_ostream< _CharT, _Traits> & operator<<  
(std::basic_ostream< _CharT, _Traits> &__os, const _Pointer_adapter< _StoreT> &__p)`
- `template<class _CharT, class _Traits, class _Alloc> basic_ostream< _CharT, _Traits> & operator<< (basic_ostream<  
_CharT, _Traits> &__o, const rope< _CharT, _Alloc> &__r)`
- `template<typename _Tp1, typename _Tp2> bool operator<= (const _Pointer_adapter< _Tp1> &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2> bool operator<= (_Tp1 __lhs, const _Pointer_adapter< _Tp2> &__rhs)`
- `template<typename _Tp1, typename _Tp2> bool operator<= (const _Pointer_adapter< _Tp1> &__lhs, const ↵  
_Pointer_adapter< _Tp2> &__rhs)`
- `template<typename _Tp> bool operator<= (const _Pointer_adapter< _Tp> &__lhs, const _Pointer_adapter< _Tp  
> &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container> bool operator<= (const __normal_iterator< ↵  
_IteratorL, _Container> &__lhs, const __normal_iterator< _IteratorR, _Container> &__rhs)`
- `template<typename _Iterator, typename _Container> bool operator<= (const __normal_iterator< _Iterator, _Container  
> &__lhs, const __normal_iterator< _Iterator, _Container> &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename> class _Base> bool  
operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base> &__lhs, const __versa_string< _CharT,  
_Traits, _Alloc, _Base> &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename> class _Base> bool  
operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base> &__lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename> class _Base> bool  
operator<= (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base> &__rhs)`
- `template<typename V, typename I, typename S> bool operator== (const character< V, I, S> &lhs, const character<  
V, I, S> &rhs)`
- `template<typename _Tp> bool operator== (const new_allocator< _Tp> &, const new_allocator< _Tp> &)`
- `template<typename _Tp> bool operator== (const malloc_allocator< _Tp> &, const malloc_allocator< _Tp> &)`
- `template<typename _Tp, typename _Array> bool operator== (const array_allocator< _Tp, _Array> &, const array_↵  
allocator< _Tp, _Array> &)`

- `template<typename _Tp> bool operator== (const \_\_pool\_alloc< _Tp > &, const \_\_pool\_alloc< _Tp > &)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All> bool operator== (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All> &_ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All> &_ht2)`
- `template<typename _Tp1, typename _Tp2> bool operator== (const Pointer\_adapter< _Tp1 > &_lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2> bool operator== (const Pointer\_adapter< _Tp1 > &_lhs, const Pointer\_adapter< _Tp2 > &_rhs)`
- `template<typename _Tp1, typename _Tp2> bool operator== (_Tp1 __lhs, const Pointer\_adapter< _Tp2 > &_rhs)`
- `template<typename _Tp> bool operator== (const Pointer\_adapter< _Tp > &_lhs, int __rhs)`
- `template<typename _Cond> bool operator== (const throw\_value\_base< _Cond > &_a, const throw\_value\_base< _Cond > &_b)`
- `template<typename _Tp> bool operator== (int __lhs, const Pointer\_adapter< _Tp > &_rhs)`
- `template<typename _Tp> bool operator== (const Pointer\_adapter< _Tp > &_lhs, const Pointer\_adapter< _Tp > &_rhs)`
- `template<typename _Tp, typename _Cond> bool operator== (const throw\_allocator\_base< _Tp, _Cond > &, const throw\_allocator\_base< _Tp, _Cond > &)`
- `template<typename _Tp, typename _Poolp> bool operator== (const \_\_mt\_alloc< _Tp, _Poolp > &, const \_\_mt\_alloc< _Tp, _Poolp > &)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container> bool operator== (const \_\_normal\_iterator< _IteratorL, _Container > &_lhs, const \_\_normal\_iterator< _IteratorR, _Container > &_rhs)`
- `template<typename _Iterator, typename _Container> bool operator== (const \_\_normal\_iterator< _Iterator, _Container > &_lhs, const \_\_normal\_iterator< _Iterator, _Container > &_rhs)`
- `template<typename _Tp1, typename _Tp2> bool operator== (const bitmap\_allocator< _Tp1 > &, const bitmap\_allocator< _Tp2 > &) throw ()`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename> class _Base> bool operator== (const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &_lhs, const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &_rhs)`
- `template<typename _CharT, template< typename, typename, typename> class _Base> \_\_enable\_if< std::is_char< _CharT >::value, bool>::type operator== (const \_\_versa\_string< _CharT, std::char\_traits< _CharT >, std::allocator< _CharT >, _Base > &_lhs, const \_\_versa\_string< _CharT, std::char\_traits< _CharT >, std::allocator< _CharT >, _Base > &_rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename> class _Base> bool operator== (const _CharT *__lhs, const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &_rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename> class _Base> bool operator== (const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &_lhs, const _CharT *__rhs)`
- `template<typename _Tp1, typename _Tp2> bool operator> (const Pointer\_adapter< _Tp1 > &_lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2> bool operator> (_Tp1 __lhs, const Pointer\_adapter< _Tp2 > &_rhs)`
- `template<typename _Tp1, typename _Tp2> bool operator> (const Pointer\_adapter< _Tp1 > &_lhs, const Pointer\_adapter< _Tp2 > &_rhs)`
- `template<typename _Tp> bool operator> (const Pointer\_adapter< _Tp > &_lhs, const Pointer\_adapter< _Tp > &_rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container> bool operator> (const \_\_normal\_iterator< _IteratorL, _Container > &_lhs, const \_\_normal\_iterator< _IteratorR, _Container > &_rhs)`
- `template<typename _Iterator, typename _Container> bool operator> (const \_\_normal\_iterator< _Iterator, _Container > &_lhs, const \_\_normal\_iterator< _Iterator, _Container > &_rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename> class _Base> bool operator> (const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &_lhs, const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &_rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename> class _Base> bool operator> (const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &_lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename> class _Base> bool operator> (const _CharT *__lhs, const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &_rhs)`
- `template<typename _Tp1, typename _Tp2> bool operator>= (_Tp1 __lhs, const Pointer\_adapter< _Tp2 > &_rhs)`



- `template<typename _Tp1 , typename _Tp2 > bool operator>= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1 , typename _Tp2 > bool operator>= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp > bool operator>= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _IteratorL , typename _IteratorR , typename _Container > bool operator>= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator , typename _Container > bool operator>= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT * __rhs)`
- `template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool operator>= (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `void rotate (_Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __first, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __middle, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __last)`
- `template<typename _Tp > void swap (_ExtPtr_allocator< _Tp > &__larg, _ExtPtr_allocator< _Tp > &__rarg)`
- `template<typename _Cond > void swap (throw_value_base< _Cond > &__a, throw_value_base< _Cond > &__b)`
- `template<class _Val , class _Key , class _HF , class _Extract , class _EqKey , class _All > void swap (hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht1, hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht2)`
- `template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> void swap (__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `_Atomic_word int throw ()`

## Variables

- `static const _Lock_policy __default_lock_policy`
- `static _Atomic_word int __val __val`

### 3.1.1 Detailed Description

GNU extensions for public use.

### 3.1.2 Function Documentation

**3.1.2.1** `template<typename _ToType , typename _FromType > _ToType __gnu_cxx::__static_pointer_cast ( const _FromType & __arg ) [inline]`

Casting operations for cases where `_FromType` is not a standard pointer. `_ToType` can be a standard or non-standard pointer. Given that `_FromType` is not a pointer, it must have a `get()` method that returns the standard pointer equivalent of the address it points to, and must have an `element_type` typedef which names the type it points to.

Definition at line 68 of file `cast.h`.

3.1.2.2 `template<typename _ToType, typename _FromType> _ToType __gnu_cxx::__static_pointer_cast( _FromType * __arg )`  
`[inline]`

Casting operations for cases where `_FromType` is a standard pointer. `_ToType` can be a standard or non-standard pointer.

Definition at line 96 of file `cast.h`.

3.1.2.3 `size_t __gnu_cxx::Bit_scan_forward( size_t __num )` `[inline]`

Generic Version of the `bsf` instruction.

Definition at line 513 of file `bitmap_allocator.h`.

Referenced by `__gnu_cxx::bitmap_allocator< _Tp >::_M_allocate_single_object()`.

3.1.2.4 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator!=( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs )` `[inline]`

Test difference of two strings.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Returns

True if `__lhs.compare(__rhs) != 0`. False otherwise.

Definition at line 2279 of file `vstring.h`.

3.1.2.5 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator!=( const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs )` `[inline]`

Test difference of C string and string.

Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

Returns

True if `__rhs.compare(__lhs) != 0`. False otherwise.

Definition at line 2292 of file `vstring.h`.

3.1.2.6 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator!=( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT * __rhs )` `[inline]`

Test difference of string and C string.

## Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

## Returns

True if `__lhs.compare(__rhs) != 0`. False otherwise.

Definition at line 2305 of file `vstring.h`.

3.1.2.7 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string<_CharT, _Traits, _Alloc, _Base> __gnu_cxx::operator+ ( const __versa_string<_CharT, _Traits, _Alloc, _Base > & __lhs, const __versa_string<_CharT, _Traits, _Alloc, _Base > & __rhs )`

Concatenate two strings.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

## Returns

New string with value of `__lhs` followed by `__rhs`.

3.1.2.8 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string<_CharT, _Traits, _Alloc, _Base> __gnu_cxx::operator+ ( const _CharT * __lhs, const __versa_string<_CharT, _Traits, _Alloc, _Base > & __rhs )`

Concatenate C string and string.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

## Returns

New string with value of `__lhs` followed by `__rhs`.

3.1.2.9 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string<_CharT, _Traits, _Alloc, _Base> __gnu_cxx::operator+ ( _CharT __lhs, const __versa_string<_CharT, _Traits, _Alloc, _Base > & __rhs )`

Concatenate character and string.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

## Returns

New string with `__lhs` followed by `__rhs`.

3.1.2.10 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string<_CharT, _Traits, _Alloc, _Base> __gnu_cxx::operator+ ( const __versa_string<_CharT, _Traits, _Alloc, _Base> & __lhs, const _CharT * __rhs )`

Concatenate string and C string.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

## Returns

New string with `__lhs` followed by `__rhs`.

3.1.2.11 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string<_CharT, _Traits, _Alloc, _Base> __gnu_cxx::operator+ ( const __versa_string<_CharT, _Traits, _Alloc, _Base> & __lhs, _CharT __rhs )`

Concatenate string and character.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

## Returns

New string with `__lhs` followed by `__rhs`.

3.1.2.12 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator< ( const __versa_string<_CharT, _Traits, _Alloc, _Base> & __lhs, const __versa_string<_CharT, _Traits, _Alloc, _Base> & __rhs ) [inline]`

Test if string precedes string.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

## Returns

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2319 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`.

3.1.2.13 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator< ( const __versa_string<_CharT, _Traits, _Alloc, _Base> & __lhs, const _CharT * __rhs ) [inline]`

Test if string precedes C string.

## Parameters

<code>__lhs</code>	String.
--------------------	---------

<code>__rhs</code>	C string.
--------------------	-----------

**Returns**

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2332 of file `vstring.h`.

**3.1.2.14** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator< ( const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]`

Test if C string precedes string.

**Parameters**

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

**Returns**

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2345 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

**3.1.2.15** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator<= ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]`

Test if string doesn't follow string.

**Parameters**

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

**Returns**

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2399 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

**3.1.2.16** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator<= ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT * __rhs ) [inline]`

Test if string doesn't follow C string.

**Parameters**

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

**Returns**

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2412 of file `vstring.h`.

```
3.1.2.17 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> bool __gnu_cxx::operator<= ( const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base >
& __rhs ) [inline]
```

Test if C string doesn't follow string.

**Parameters**

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

**Returns**

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2425 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

```
3.1.2.18 template<typename _Tp > bool __gnu_cxx::operator==( const _Pointer_adapter< _Tp > & __lhs, const
_PPointer_adapter< _Tp > & __rhs ) [inline]
```

Comparison operators for `_Pointer_adapter` defer to the base class' comparison operators, when possible.

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

```
3.1.2.19 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> bool __gnu_cxx::operator==( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const
__versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]
```

Test equivalence of two strings.

**Parameters**

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

**Returns**

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 2228 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

```
3.1.2.20 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> bool __gnu_cxx::operator==( const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &
__rhs ) [inline]
```

Test equivalence of C string and string.

## Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

## Returns

True if `__rhs.compare(__lhs) == 0`. False otherwise.

Definition at line 2252 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

**3.1.2.21** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator==( const __versa_string<_CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT * __rhs ) [inline]`

Test equivalence of string and C string.

## Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

## Returns

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 2265 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

**3.1.2.22** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator> ( const __versa_string<_CharT, _Traits, _Alloc, _Base > & __lhs, const __versa_string<_CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]`

Test if string follows string.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

## Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2359 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

**3.1.2.23** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator> ( const __versa_string<_CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT * __rhs ) [inline]`

Test if string follows C string.



## Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

## Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2372 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

3.1.2.24 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator> ( const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]`

Test if C string follows string.

## Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

## Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2385 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

3.1.2.25 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator>= ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]`

Test if string doesn't precede string.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

## Returns

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2439 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

3.1.2.26 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator>= ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT * __rhs ) [inline]`

Test if string doesn't precede C string.

## Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

## Returns

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2452 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`.

```
3.1.2.27 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> bool __gnu_cxx::operator>= ( const _CharT* __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base >
& __rhs ) [inline]
```

Test if C string doesn't precede string.

## Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

## Returns

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2465 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`.

```
3.1.2.28 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> void __gnu_cxx::swap ( __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, __versa_string<
_CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]
```

Swap contents of two strings.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Exchanges the contents of `__lhs` and `__rhs` in constant time.

Definition at line 2479 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::swap()`.

3.2 `__gnu_cxx::__detail` Namespace Reference

## Classes

- class [\\_\\_mini\\_vector](#)
- class [\\_Bitmap\\_counter](#)
- class [\\_Ffit\\_finder](#)

## Enumerations

- enum { **bits\_per\_byte**, **bits\_per\_block** }

## Functions

- void **\_\_bit\_allocate** (size\_t \*\_\_pmap, size\_t \_\_pos) throw ()
- void **\_\_bit\_free** (size\_t \*\_\_pmap, size\_t \_\_pos) throw ()
- template<typename \_ForwardIterator, typename \_Tp, typename \_Compare > \_ForwardIterator **\_\_lower\_bound** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp &\_\_val, \_Compare \_\_comp)
- template<typename \_AddrPair > size\_t **\_\_num\_bitmaps** (\_AddrPair \_\_ap)
- template<typename \_AddrPair > size\_t **\_\_num\_blocks** (\_AddrPair \_\_ap)

### 3.2.1 Detailed Description

Implementation details not part of the namespace `__gnu_cxx` interface.

### 3.2.2 Function Documentation

**3.2.2.1** void `__gnu_cxx::detail::__bit_allocate ( size_t * __pmap, size_t __pos ) throw ()` [inline]

Mark a memory address as allocated by re-setting the corresponding bit in the bit-map.

Definition at line 488 of file `bitmap_allocator.h`.

Referenced by `__gnu_cxx::bitmap_allocator<_Tp>::_M_allocate_single_object()`.

**3.2.2.2** void `__gnu_cxx::detail::__bit_free ( size_t * __pmap, size_t __pos ) throw ()` [inline]

Mark a memory address as free by setting the corresponding bit in the bit-map.

Definition at line 499 of file `bitmap_allocator.h`.

Referenced by `__gnu_cxx::bitmap_allocator<_Tp>::_M_deallocate_single_object()`.

**3.2.2.3** template<typename \_AddrPair > size\_t `__gnu_cxx::detail::__num_bitmaps ( _AddrPair __ap )` [inline]

The number of Bit-maps pointed to by the address pair passed to the function.

Definition at line 276 of file `bitmap_allocator.h`.

References `__num_blocks()`.

Referenced by `__gnu_cxx::bitmap_allocator<_Tp>::_M_allocate_single_object()`, and `__gnu_cxx::bitmap_allocator<_Tp>::_M_deallocate_single_object()`.

**3.2.2.4** template<typename \_AddrPair > size\_t `__gnu_cxx::detail::__num_blocks ( _AddrPair __ap )` [inline]

The number of Blocks pointed to by the address pair passed to the function.

Definition at line 268 of file `bitmap_allocator.h`.

Referenced by `__num_bitmaps()`.

## 3.3 `__gnu_cxx::typelist` Namespace Reference

## Functions

- `template<typename Fn, typename Typelist> void apply (Fn &, Typelist)`
- `template<typename Gn, typename Typelist> void apply_generator (Gn &, Typelist)`
- `template<typename Gn, typename TypelistT, typename TypelistV> void apply_generator (Gn &, TypelistT, TypelistV)`
- `template<typename Fn, typename Typelist> void apply_generator (Fn &fn, Typelist)`
- `template<typename Fn, typename TypelistT, typename TypelistV> void apply_generator (Fn &fn, TypelistT, TypelistV)`

## 3.3.1 Detailed Description

GNU typelist extensions for public compile-time use.

## 3.3.2 Function Documentation

3.3.2.1 `template<typename Gn, typename Typelist> void __gnu_cxx::typelist::apply_generator ( Gn &, Typelist )`

Apply all typelist types to generator functor.

3.4 `__gnu_debug` Namespace Reference

## Classes

- class `_After_nth_from`
- struct `_BeforeBeginHelper`
- class `_Equal_to`
- class `_Not_equal_to`
- class `_Safe_iterator`
- class `_Safe_iterator_base`
- class `_Safe_local_iterator`
- class `_Safe_local_iterator_base`
- class `_Safe_sequence`
- class `_Safe_sequence_base`
- class `_Safe_unordered_container`
- class `_Safe_unordered_container_base`

## Enumerations

- enum `_Debug_msg_id` { `__msg_valid_range`, `__msg_insert_singular`, `__msg_insert_different`, `__msg_erase_bad`, `__msg_erase_different`, `__msg_subscript_oob`, `__msg_empty`, `__msg_unpartitioned`, `__msg_unpartitioned_pred`, `__msg_unsorted`, `__msg_unsorted_pred`, `__msg_not_heap`, `__msg_not_heap_pred`, `__msg_bad_bitset_write`, `__msg_bad_bitset_read`, `__msg_bad_bitset_flip`, `__msg_self_splice`, `__msg_splice_alloc`, `__msg_splice_bad`, `__msg_splice_other`, `__msg_splice_overlap`, `__msg_init_singular`, `__msg_init_copy_singular`, `__msg_init_const_singular`, `__msg_copy_singular`, `__msg_bad_deref`, `__msg_bad_inc`, `__msg_bad_dec`, `__msg_iter_subscript_oob`, `__msg_advance_oob`, `__msg_retreat_oob`, `__msg_iter_compare_bad`, `__msg_compare_different`, `__msg_iter_order_bad`, `__msg_order_different`, `__msg_distance_bad`, `__msg_distance_different`, `__msg_deref_istream`, `__msg_inc_istream`, `__msg_output_ostream`, `__msg_deref_istreambuf`, `__msg_inc_istreambuf`, `__msg_insert_after_end`, `__msg_erase_after_bad`, `__msg_valid_range2`, `__msg_local_iter_compare_bad`, `__msg_non_empty_range`, `__msg_self_move_assign`, `__msg_bucket_index_oob`, `__msg_valid_load_factor`, `__msg_equal_allocs` }
- enum `_Distance_precision` { `__dp_equality`, `__dp_sign`, `__dp_exact` }

## Functions

- `template<typename _Iterator> _Siter_base<_Iterator>::iterator_type __base (_Iterator __it)`
- `template<typename _Iterator> bool __check_dereferenceable (_Iterator &)`
- `template<typename _Tp> bool __check_dereferenceable (const _Tp * __ptr)`
- `template<typename _Iterator, typename _Sequence> bool __check_dereferenceable (const _Safe_iterator<_Iterator, _Sequence> & __x)`
- `template<typename _ForwardIterator, typename _Tp> bool __check_partitioned_lower (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __value)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred> bool __check_partitioned_lower (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __value, _Pred __pred)`
- `template<typename _ForwardIterator, typename _Tp> bool __check_partitioned_upper (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __value)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred> bool __check_partitioned_upper (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __value, _Pred __pred)`
- `template<typename _Iterator> bool __check_singular (_Iterator &)`
- `template<typename _Tp> bool __check_singular (const _Tp * __ptr)`
- `template<typename _Iterator, typename _Sequence> bool __check_singular (const _Safe_iterator<_Iterator, _Sequence> & __x)`
- `bool __check_singular_aux (const void *)`
- `bool __check_singular_aux (const _Safe_iterator_base * __x)`
- `template<typename _InputIterator> bool __check_sorted (const _InputIterator & __first, const _InputIterator & __last)`
- `template<typename _InputIterator, typename _Predicate> bool __check_sorted (const _InputIterator & __first, const _InputIterator & __last, _Predicate __pred)`
- `template<typename _InputIterator> bool __check_sorted_aux (const _InputIterator &, const _InputIterator &, std::input_iterator_tag)`
- `template<typename _ForwardIterator> bool __check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
- `template<typename _Iterator, typename _Sequence> bool __check_sorted_aux (const _Safe_iterator<_Iterator, _Sequence> & __first, const _Safe_iterator<_Iterator, _Sequence> & __last, std::random_access_iterator_tag __tag)`
- `template<typename _InputIterator, typename _Predicate> bool __check_sorted_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::input_iterator_tag)`
- `template<typename _ForwardIterator, typename _Predicate> bool __check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, std::forward_iterator_tag)`
- `template<typename _Iterator, typename _Sequence, typename _Predicate> bool __check_sorted_aux (const _Safe_iterator<_Iterator, _Sequence> & __first, const _Safe_iterator<_Iterator, _Sequence> & __last, _Predicate __pred, std::random_access_iterator_tag __tag)`
- `template<typename _InputIterator1, typename _InputIterator2> bool __check_sorted_set (const _InputIterator1 & __first, const _InputIterator1 & __last, const _InputIterator2 &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Predicate> bool __check_sorted_set (const _InputIterator1 & __first, const _InputIterator1 & __last, const _InputIterator2 &, _Predicate __pred)`
- `template<typename _InputIterator> bool __check_sorted_set_aux (const _InputIterator & __first, const _InputIterator & __last, std::true_type)`
- `template<typename _InputIterator> bool __check_sorted_set_aux (const _InputIterator &, const _InputIterator &, std::false_type)`
- `template<typename _InputIterator, typename _Predicate> bool __check_sorted_set_aux (const _InputIterator & __first, const _InputIterator & __last, _Predicate __pred, std::true_type)`
- `template<typename _InputIterator, typename _Predicate> bool __check_sorted_set_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::false_type)`
- `template<typename _CharT, typename _Integer> const _CharT * __check_string (const _CharT * __s, const _Integer & __n __attribute__((__unused__)))`

- `template<typename _CharT> const _CharT * \_\_check\_string (const _CharT * __s)`
- `template<typename _InputIterator> _InputIterator \_\_check\_valid\_range (const _InputIterator & __first, const _InputIterator & __last, \_\_attribute\_\_\(\(\_\_unused\_\_\)\))`
- `template<typename _Iterator1, typename _Iterator2> std::pair< typename std::iterator_traits< _Iterator1 >::difference_type, \_Distance\_precision > \_\_get\_distance (const _Iterator1 & __lhs, const _Iterator2 & __rhs, std::random\_access\_iterator\_tag)`
- `template<typename _Iterator1, typename _Iterator2> std::pair< typename std::iterator_traits< _Iterator1 >::difference_type, \_Distance\_precision > \_\_get\_distance (const _Iterator1 & __lhs, const _Iterator2 & __rhs, std::forward\_iterator\_tag)`
- `template<typename _Iterator1, typename _Iterator2> std::pair< typename std::iterator_traits< _Iterator1 >::difference_type, \_Distance\_precision > \_\_get\_distance (const _Iterator1 & __lhs, const _Iterator2 & __rhs)`
- `template<typename _InputIterator> bool \_\_valid\_range (const _InputIterator & __first, const _InputIterator & __last)`
- `template<typename _Iterator, typename _Sequence> bool \_\_valid\_range (const \_Safe\_iterator< _Iterator, _Sequence > & __first, const \_Safe\_iterator< _Iterator, _Sequence > & __last)`
- `template<typename _Iterator, typename _Sequence> bool \_\_valid\_range (const \_Safe\_local\_iterator< _Iterator, _Sequence > & __first, const \_Safe\_local\_iterator< _Iterator, _Sequence > & __last)`
- `template<typename _Integral> bool \_\_valid\_range\_aux (const _Integral &, const _Integral &, std::__true_type)`
- `template<typename _InputIterator> bool \_\_valid\_range\_aux (const _InputIterator & __first, const _InputIterator & __last, std::__false_type)`
- `template<typename _RandomAccessIterator> bool \_\_valid\_range\_aux2 (const _RandomAccessIterator & __first, const _RandomAccessIterator & __last, std::random\_access\_iterator\_tag)`
- `template<typename _InputIterator> bool \_\_valid\_range\_aux2 (const _InputIterator &, const _InputIterator &, std::input\_iterator\_tag)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence> bool operator!= (const \_Safe\_local\_iterator< _IteratorL, _Sequence > & __lhs, const \_Safe\_local\_iterator< _IteratorR, _Sequence > & __rhs)`
- `template<typename _Iterator, typename _Sequence> bool operator!= (const \_Safe\_local\_iterator< _Iterator, _Sequence > & __lhs, const \_Safe\_local\_iterator< _Iterator, _Sequence > & __rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence> bool operator!= (const \_Safe\_iterator< _IteratorL, _Sequence > & __lhs, const \_Safe\_iterator< _IteratorR, _Sequence > & __rhs)`
- `template<typename _Iterator, typename _Sequence> bool operator!= (const \_Safe\_iterator< _Iterator, _Sequence > & __lhs, const \_Safe\_iterator< _Iterator, _Sequence > & __rhs)`
- `template<typename _Iterator, typename _Sequence> \_Safe\_iterator< _Iterator, _Sequence > operator+ (typename \_Safe\_iterator< _Iterator, _Sequence >::difference_type __n, const \_Safe\_iterator< _Iterator, _Sequence > & __i)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence> \_Safe\_iterator< _IteratorL, _Sequence >::difference_type operator- (const \_Safe\_iterator< _IteratorL, _Sequence > & __lhs, const \_Safe\_iterator< _IteratorR, _Sequence > & __rhs)`
- `template<typename _Iterator, typename _Sequence> \_Safe\_iterator< _Iterator, _Sequence >::difference_type operator- (const \_Safe\_iterator< _Iterator, _Sequence > & __lhs, const \_Safe\_iterator< _Iterator, _Sequence > & __rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence> bool operator< (const \_Safe\_iterator< _IteratorL, _Sequence > & __lhs, const \_Safe\_iterator< _IteratorR, _Sequence > & __rhs)`
- `template<typename _Iterator, typename _Sequence> bool operator< (const \_Safe\_iterator< _Iterator, _Sequence > & __lhs, const \_Safe\_iterator< _Iterator, _Sequence > & __rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence> bool operator<= (const \_Safe\_iterator< _IteratorL, _Sequence > & __lhs, const \_Safe\_iterator< _IteratorR, _Sequence > & __rhs)`
- `template<typename _Iterator, typename _Sequence> bool operator<= (const \_Safe\_iterator< _Iterator, _Sequence > & __lhs, const \_Safe\_iterator< _Iterator, _Sequence > & __rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence> bool operator== (const \_Safe\_local\_iterator< _IteratorL, _Sequence > & __lhs, const \_Safe\_local\_iterator< _IteratorR, _Sequence > & __rhs)`
- `template<typename _Iterator, typename _Sequence> bool operator== (const \_Safe\_local\_iterator< _Iterator, _Sequence > & __lhs, const \_Safe\_local\_iterator< _Iterator, _Sequence > & __rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence> bool operator== (const \_Safe\_iterator< _IteratorL, _Sequence > & __lhs, const \_Safe\_iterator< _IteratorR, _Sequence > & __rhs)`

- `template<typename _Iterator, typename _Sequence> bool operator== (const \_Safe\_iterator< _Iterator, _Sequence > &__lhs, const \_Safe\_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence> bool operator> (const \_Safe\_iterator< _IteratorL, _Sequence > &__lhs, const \_Safe\_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence> bool operator> (const \_Safe\_iterator< _Iterator, _Sequence > &__lhs, const \_Safe\_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence> bool operator>= (const \_Safe\_iterator< _IteratorL, _Sequence > &__lhs, const \_Safe\_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence> bool operator>= (const \_Safe\_iterator< _Iterator, _Sequence > &__lhs, const \_Safe\_iterator< _Iterator, _Sequence > &__rhs)`

### 3.4.1 Detailed Description

GNU debug classes for public use.

### 3.4.2 Enumeration Type Documentation

#### 3.4.2.1 `enum __gnu_debug::__Distance_precision`

The precision to which we can calculate the distance between two iterators.

Definition at line 70 of file `safe_iterator.h`.

### 3.4.3 Function Documentation

#### 3.4.3.1 `template<typename _Iterator> _Siter_base<_Iterator>::iterator_type __gnu_debug::__base ( _Iterator __it )` [inline]

Helper function to extract base iterator of random access safe iterator in order to reduce performance impact of debug mode. Limited to random access iterator because it is the only category for which it is possible to check for correct iterators order in the `__valid_range` function thanks to the `<` operator.

Definition at line 446 of file `functions.h`.

Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`, `__gnu_debug::__Safe_iterator< _Iterator, _Sequence >::__M_before_dereferenceable()`, `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, `std::nouppercase()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

#### 3.4.3.2 `template<typename _Iterator> bool __gnu_debug::__check_dereferenceable ( _Iterator & )` [inline]

Assume that some arbitrary iterator is dereferenceable, because we can't prove that it isn't.

Definition at line 70 of file `functions.h`.

#### 3.4.3.3 `template<typename _Tp> bool __gnu_debug::__check_dereferenceable ( const _Tp * __ptr )` [inline]

Non-NULL pointers are dereferenceable.

Definition at line 76 of file `functions.h`.

#### 3.4.3.4 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::__check_dereferenceable ( const \_Safe\_iterator< _Iterator, _Sequence > & __x )` [inline]

Safe iterators know if they are singular.

Definition at line 82 of file functions.h.

References \_\_gnu\_debug::\_\_Safe\_iterator<\_Iterator, \_Sequence>::\_\_M\_dereferenceable().

3.4.3.5 `template<typename _Tp> bool __gnu_debug::__check_singular ( const _Tp* __ptr ) [inline]`

Non-NULL pointers are nonsingular.

Definition at line 57 of file functions.h.

3.4.3.6 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::__check_singular ( const _Safe_iterator<_Iterator, _Sequence> & __x ) [inline]`

Safe iterators know if they are singular.

Definition at line 63 of file functions.h.

References \_\_gnu\_debug::\_\_Safe\_iterator\_base::\_\_M\_singular().

3.4.3.7 `bool __gnu_debug::__check_singular_aux ( const _Safe_iterator_base* __x ) [inline]`

Iterators that derive from \_Safe\_iterator\_base but that aren't \_Safe\_iterators can be determined singular or non-singular via \_Safe\_iterator\_base.

Definition at line 64 of file safe\_iterator.h.

References \_\_gnu\_debug::\_\_Safe\_iterator\_base::\_\_M\_singular().

3.4.3.8 `template<typename _CharT, typename _Integer> const _CharT* __gnu_debug::__check_string ( const _CharT* __s, const _Integer & __n __attribute__( __unused__ ) ) [inline]`

Checks that \_\_s is non-NULL or \_\_n == 0, and then returns \_\_s.

Definition at line 168 of file functions.h.

3.4.3.9 `template<typename _CharT> const _CharT* __gnu_debug::__check_string ( const _CharT* __s ) [inline]`

Checks that \_\_s is non-NULL and then returns \_\_s.

Definition at line 180 of file functions.h.

3.4.3.10 `template<typename _Iterator1, typename _Iterator2> std::pair<typename std::iterator_traits<_Iterator1>::difference_type, _Distance_precision> __gnu_debug::__get_distance ( const _Iterator1 & __lhs, const _Iterator2 & __rhs, std::random_access_iterator_tag ) [inline]`

Determine the distance between two iterators with some known precision.

Definition at line 83 of file safe\_iterator.h.

References std::make\_pair().

3.4.3.11 `template<typename _InputIterator> bool __gnu_debug::__valid_range ( const _InputIterator & __first, const _InputIterator & __last ) [inline]`

Don't know what these iterators are, or if they are even iterators (we may get an integral type for InputIterator), so see if they are integral and pass them on to the next phase otherwise.

Definition at line 131 of file functions.h.

References \_\_valid\_range\_aux().



3.4.3.12 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::__valid_range ( const __Safe_iterator<_Iterator, _Sequence> & __first, const __Safe_iterator<_Iterator, _Sequence> & __last ) [inline]`

Safe iterators know how to check if they form a valid range.

Definition at line 140 of file functions.h.

3.4.3.13 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::__valid_range ( const __Safe_local_iterator<_Iterator, _Sequence> & __first, const __Safe_local_iterator<_Iterator, _Sequence> & __last ) [inline]`

Safe local iterators know how to check if they form a valid range.

Definition at line 147 of file functions.h.

3.4.3.14 `template<typename _Integral> bool __gnu_debug::__valid_range_aux ( const _Integral &, const _Integral &, std::__true_type ) [inline]`

We say that integral types for a valid range, and defer to other routines to realize what to do with integral types instead of iterators.

Definition at line 111 of file functions.h.

Referenced by `__valid_range()`.

3.4.3.15 `template<typename _InputIterator> bool __gnu_debug::__valid_range_aux ( const _InputIterator & __first, const _InputIterator & __last, std::__false_type ) [inline]`

We have iterators, so figure out what kind of iterators that are to see if we can check the range ahead of time.

Definition at line 119 of file functions.h.

References `std::__iterator_category()`, and `__valid_range_aux2()`.

3.4.3.16 `template<typename _RandomAccessIterator> bool __gnu_debug::__valid_range_aux2 ( const _RandomAccessIterator & __first, const _RandomAccessIterator & __last, std::random_access_iterator_tag ) [inline]`

If the distance between two random access iterators is nonnegative, assume the range is valid.

Definition at line 90 of file functions.h.

Referenced by `__valid_range_aux()`.

3.4.3.17 `template<typename _InputIterator> bool __gnu_debug::__valid_range_aux2 ( const _InputIterator &, const _InputIterator &, std::input_iterator_tag ) [inline]`

Can't test for a valid range with input iterators, because iteration may be destructive. So we just assume that the range is valid.

Definition at line 101 of file functions.h.

## 3.5 `__gnu_internal` Namespace Reference

### 3.5.1 Detailed Description

GNU implementation details, not for public use or export. Used only when anonymous namespaces cannot be substituted.

3.6 `__gnu_parallel` Namespace Reference

## Classes

- struct [\\_\\_accumulate\\_binop\\_reduct](#)
- struct [\\_\\_accumulate\\_selector](#)
- struct [\\_\\_adjacent\\_difference\\_selector](#)
- struct [\\_\\_adjacent\\_find\\_selector](#)
- class [\\_\\_binder1st](#)
- class [\\_\\_binder2nd](#)
- struct [\\_\\_count\\_if\\_selector](#)
- struct [\\_\\_count\\_selector](#)
- struct [\\_\\_fill\\_selector](#)
- struct [\\_\\_find\\_first\\_of\\_selector](#)
- struct [\\_\\_find\\_if\\_selector](#)
- struct [\\_\\_for\\_each\\_selector](#)
- struct [\\_\\_generate\\_selector](#)
- struct [\\_\\_generic\\_find\\_selector](#)
- struct [\\_\\_generic\\_for\\_each\\_selector](#)
- struct [\\_\\_identity\\_selector](#)
- struct [\\_\\_inner\\_product\\_selector](#)
- struct [\\_\\_max\\_element\\_reduct](#)
- struct [\\_\\_min\\_element\\_reduct](#)
- struct [\\_\\_mismatch\\_selector](#)
- struct [\\_\\_multiway\\_merge\\_3\\_variant\\_sentinel\\_switch](#)
- struct [\\_\\_multiway\\_merge\\_3\\_variant\\_sentinel\\_switch< true, \\_RAIterIterator, \\_RAIter3, \\_DifferenceTp, \\_Compare >](#)
- struct [\\_\\_multiway\\_merge\\_4\\_variant\\_sentinel\\_switch](#)
- struct [\\_\\_multiway\\_merge\\_4\\_variant\\_sentinel\\_switch< true, \\_RAIterIterator, \\_RAIter3, \\_DifferenceTp, \\_Compare >](#)
- struct [\\_\\_multiway\\_merge\\_k\\_variant\\_sentinel\\_switch](#)
- struct [\\_\\_multiway\\_merge\\_k\\_variant\\_sentinel\\_switch< false, \\_\\_stable, \\_RAIterIterator, \\_RAIter3, \\_DifferenceTp, \\_Compare >](#)
- struct [\\_\\_replace\\_if\\_selector](#)
- struct [\\_\\_replace\\_selector](#)
- struct [\\_\\_transform1\\_selector](#)
- struct [\\_\\_transform2\\_selector](#)
- class [\\_\\_unary\\_negate](#)
- struct [\\_DRandomShufflingGlobalData](#)
- struct [\\_DRSSorterPU](#)
- struct [\\_DummyReduct](#)
- class [\\_EqualFromLess](#)
- struct [\\_EqualTo](#)
- class [\\_GuardedIterator](#)
- class [\\_IteratorPair](#)
- class [\\_IteratorTriple](#)
- struct [\\_Job](#)
- struct [\\_Less](#)
- class [\\_Lexicographic](#)
- class [\\_LexicographicReverse](#)
- class [\\_LoserTree](#)

- class [\\_LoserTree< false, \\_Tp, \\_Compare >](#)
- class [\\_LoserTreeBase](#)
- class [\\_LoserTreePointer](#)
- class [\\_LoserTreePointer< false, \\_Tp, \\_Compare >](#)
- class [\\_LoserTreePointerBase](#)
- class [\\_LoserTreePointerUnguarded](#)
- class [\\_LoserTreePointerUnguarded< false, \\_Tp, \\_Compare >](#)
- class [\\_LoserTreePointerUnguardedBase](#)
- struct [\\_LoserTreeTraits](#)
- class [\\_LoserTreeUnguarded](#)
- class [\\_LoserTreeUnguarded< false, \\_Tp, \\_Compare >](#)
- class [\\_LoserTreeUnguardedBase](#)
- struct [\\_Multiplies](#)
- struct [\\_Nothing](#)
- struct [\\_Piece](#)
- struct [\\_Plus](#)
- struct [\\_PMWMSSortingData](#)
- class [\\_PseudoSequence](#)
- class [\\_PseudoSequenceIterator](#)
- struct [\\_QSBThreadLocal](#)
- class [\\_RandomNumber](#)
- class [\\_RestrictedBoundedConcurrentQueue](#)
- struct [\\_SamplingSorter](#)
- struct [\\_SamplingSorter< false, \\_RAIter, \\_StrictWeakOrdering >](#)
- struct [\\_Settings](#)
- struct [\\_SplitConsistently](#)
- struct [\\_SplitConsistently< false, \\_RAIter, \\_Compare, \\_SortingPlacesIterator >](#)
- struct [\\_SplitConsistently< true, \\_RAIter, \\_Compare, \\_SortingPlacesIterator >](#)
- struct [balanced\\_quicksort\\_tag](#)
- struct [balanced\\_tag](#)
- struct [constant\\_size\\_blocks\\_tag](#)
- struct [default\\_parallel\\_tag](#)
- struct [equal\\_split\\_tag](#)
- struct [exact\\_tag](#)
- struct [find\\_tag](#)
- struct [growing\\_blocks\\_tag](#)
- struct [multiway\\_mergesort\\_exact\\_tag](#)
- struct [multiway\\_mergesort\\_sampling\\_tag](#)
- struct [multiway\\_mergesort\\_tag](#)
- struct [omp\\_loop\\_static\\_tag](#)
- struct [omp\\_loop\\_tag](#)
- struct [parallel\\_tag](#)
- struct [quicksort\\_tag](#)
- struct [sampling\\_tag](#)
- struct [sequential\\_tag](#)
- struct [unbalanced\\_tag](#)

## Typedefs

- typedef unsigned short `_BinIndex`
- typedef int64\_t `_CASable`
- typedef uint64\_t `_SequenceIndex`
- typedef uint16\_t `_ThreadIndex`

## Enumerations

- enum `_AlgorithmStrategy` { `heuristic`, `force_sequential`, `force_parallel` }
- enum `_FindAlgorithm` { `GROWING_BLOCKS`, `CONSTANT_SIZE_BLOCKS`, `EQUAL_SPLIT` }
- enum `_MultiwayMergeAlgorithm` { `LOSER_TREE` }
- enum `_Parallelism` { `sequential`, `parallel_unbalanced`, `parallel_balanced`, `parallel_omp_loop`, `parallel_omp_loop_↵_static`, `parallel_taskqueue` }
- enum `_PartialSumAlgorithm` { `RECURSIVE`, `LINEAR` }
- enum `_SortAlgorithm` { `MWMS`, `QS`, `QS_BALANCED` }
- enum `_SplittingAlgorithm` { `SAMPLING`, `EXACT` }

## Functions

- template<typename \_Tp> `_Tp __add_omp` (volatile \_Tp \* \_\_ptr, \_Tp \_\_addend)
- template<typename \_RAIter, typename \_DifferenceTp> void `__calc_borders` (\_RAIter \_\_elements, \_DifferenceTp \_\_↵length, \_DifferenceTp \* \_\_off)
- template<typename \_Tp> bool `__cas_omp` (volatile \_Tp \* \_\_ptr, \_Tp \_\_comparand, \_Tp \_\_replacement)
- template<typename \_Tp> bool `__compare_and_swap` (volatile \_Tp \* \_\_ptr, \_Tp \_\_comparand, \_Tp \_\_replacement)
- template<typename \_IIter, typename \_OutputIterator> \_OutputIterator `__copy_tail` (std::pair< \_IIter, \_IIter> \_\_b, std↵::pair< \_IIter, \_IIter> \_\_e, \_OutputIterator \_\_r)
- void `__decode2` (\_CASable \_\_x, int & \_\_a, int & \_\_b)
- template<typename \_RAIter, typename \_DifferenceTp> void `__determine_samples` (\_PMWSSortingData< \_RAIter> \* \_\_sd, \_DifferenceTp \_\_num\_samples)
- `_CASable __encode2` (int \_\_a, int \_\_b)
- template<typename \_DifferenceType, typename \_OutputIterator> \_OutputIterator `__equally_split` (\_DifferenceType \_\_n, ↵\_ThreadIndex \_\_num\_threads, \_OutputIterator \_\_s)
- template<typename \_DifferenceType> \_DifferenceType `__equally_split_point` (\_DifferenceType \_\_n, \_ThreadIndex ↵\_\_num\_threads, \_ThreadIndex \_\_thread\_no)
- template<typename \_Tp> \_Tp `__fetch_and_add` (volatile \_Tp \* \_\_ptr, \_Tp \_\_addend)
- template<typename \_RAIter1, typename \_RAIter2, typename \_Pred, typename \_Selector> std::pair< \_RAIter1, \_RAIter2> ↵`__find_template` (\_RAIter1 \_\_begin1, \_RAIter1 \_\_end1, \_RAIter2 \_\_begin2, \_Pred \_\_pred, \_Selector \_\_selector)
- template<typename \_RAIter1, typename \_RAIter2, typename \_Pred, typename \_Selector> std::pair< \_RAIter1, \_RAIter2> ↵`__find_template` (\_RAIter1 \_\_begin1, \_RAIter1 \_\_end1, \_RAIter2 \_\_begin2, \_Pred \_\_pred, \_Selector \_\_selector, ↵`equal_split_tag`)
- template<typename \_RAIter1, typename \_RAIter2, typename \_Pred, typename \_Selector> std::pair< \_RAIter1, \_RAIter2> ↵`__find_template` (\_RAIter1 \_\_begin1, \_RAIter1 \_\_end1, \_RAIter2 \_\_begin2, \_Pred \_\_pred, \_Selector \_\_selector, ↵`growing_blocks_tag`)
- template<typename \_RAIter1, typename \_RAIter2, typename \_Pred, typename \_Selector> std::pair< \_RAIter1, \_RAIter2> ↵`__find_template` (\_RAIter1 \_\_begin1, \_RAIter1 \_\_end1, \_RAIter2 \_\_begin2, \_Pred \_\_pred, \_Selector \_\_selector, ↵`constant_size_blocks_tag`)
- template<typename \_IIter, typename \_UserOp, typename \_Functionality, typename \_Red, typename \_Result> \_UserOp ↵`__for_each_template_random_access` (\_IIter \_\_begin, \_IIter \_\_end, \_UserOp \_\_user\_op, \_Functionality & \_\_↵functionality, \_Red \_\_reduction, \_Result \_\_reduction\_start, \_Result & \_\_output, typename std::iterator\_traits< ↵\_IIter>::difference\_type \_\_bound, \_Parallelism \_\_parallelism\_tag)

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result> _Op \_\_for\_each\_template↵  
\_\_random\_access\_ed (_RAIter __begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r, _Result __base, _Result  
& __output, typename std::iterator_traits< _RAIter >::difference_type __bound)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result> _Op \_\_for\_each\_template↵  
\_\_random\_access\_omp\_loop (_RAIter __begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r, _Result __base,  
_Result & __output, typename std::iterator_traits< _RAIter >::difference_type __bound)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result> _Op \_\_for\_each\_template↵  
\_\_random\_access\_omp\_loop\_static (_RAIter __begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r, _Result  
__base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type __bound)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result> _Op \_\_for\_each\_template↵  
\_\_random\_access\_workstealing (_RAIter __begin, _RAIter __end, _Op __op, _Fu & __f, _Red __r, _Result __base,  
_Result & __output, typename std::iterator_traits< _RAIter >::difference_type __bound)`
- `\_ThreadIndex \_\_get\_max\_threads ()`
- `bool \_\_is\_parallel (const \_Parallelism __p)`
- `template<typename _Iter, typename _Compare> bool \_\_is\_sorted (_Iter __begin, _Iter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare> _RAIter \_\_median\_of\_three\_iterators (_RAIter __a, _RAIter __b, ↵  
_RAIter __c, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare> ↵  
_OutputIterator \_\_merge\_advance (_RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2 ↵  
__end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare> ↵  
_OutputIterator \_\_merge\_advance\_movc (_RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2 ↵  
__end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare> ↵  
_OutputIterator \_\_merge\_advance\_usual (_RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2 ↵  
__end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _Compare> _RAIter3 \_\_parallel\_merge↵  
\_\_advance (_RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2 __end2, _RAIter3 __target,  
typename std::iterator_traits< _RAIter1 >::difference_type __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter3, typename _Compare> _RAIter3 \_\_parallel\_merge\_advance (_RAIter1  
& __begin1, _RAIter1 __end1, _RAIter1 & __begin2, _RAIter1 __end2, _RAIter3 __target, typename std::iterator↵  
_traits< _RAIter1 >::difference_type __max_length, _Compare __comp)`
- `template<typename _RAIter, typename _Compare> void \_\_parallel\_nth\_element (_RAIter __begin, _RAIter __nth, ↵  
_RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare> void \_\_parallel\_partial\_sort (_RAIter __begin, _RAIter __middle, ↵  
_RAIter __end, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation> _OutputIterator \_\_parallel\_partial\_sum (↵  
_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation> _OutputIterator \_\_parallel\_partial\_sum↵  
\_\_basecase (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename std::↵  
_iterator_traits< _Iter >::value_type __value)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation> _OutputIterator \_\_parallel\_partial\_sum↵  
\_\_linear (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator↵  
_traits< _Iter >::difference_type __n)`
- `template<typename _RAIter, typename _Predicate> std::iterator_traits< _RAIter >::difference_type \_\_parallel\_partition  
( _RAIter __begin, _RAIter __end, _Predicate __pred, \_ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _RandomNumberGenerator> void \_\_parallel\_random\_shuffle (_RAIter __begin, ↵  
_RAIter __end, _RandomNumberGenerator __rng=\_RandomNumber())`
- `template<typename _RAIter, typename _RandomNumberGenerator> void \_\_parallel\_random\_shuffle\_drs (_RAIter __begin,  
_RAIter __end, typename std::iterator_traits< _RAIter >::difference_type __n, \_ThreadIndex __num_threads,  
_RandomNumberGenerator & __rng)`
- `template<typename _RAIter, typename _RandomNumberGenerator> void \_\_parallel\_random\_shuffle\_drs\_pu (_DRS↵  
SorterPU< _RAIter, _RandomNumberGenerator > * __pus)`

- `template<typename _Iter, typename _OutputIterator, typename _Compare> _OutputIterator \_\_parallel\_set\_difference (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare> _OutputIterator \_\_parallel\_set\_intersection (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Operation> _OutputIterator \_\_parallel\_set\_operation (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Operation __op)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare> _OutputIterator \_\_parallel\_set\_symmetric\_difference (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare> _OutputIterator \_\_parallel\_set\_union (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<bool __stable, typename _RAIter, typename _Compare, typename _Parallelism> void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_Parallelism __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare> void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway\_mergesort\_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare> void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway\_mergesort\_exact\_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare> void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway\_mergesort\_sampling\_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare> void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, quicksort\_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare> void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, balanced\_quicksort\_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare> void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, default\_parallel\_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare> void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, parallel\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare> void \_\_parallel\_sort\_qs (_RAIter __begin, _RAIter __end, _Compare __comp, \_ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare> void \_\_parallel\_sort\_qs\_conquer (_RAIter __begin, _RAIter __end, _Compare __comp, \_ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare> std::iterator_traits<_RAIter>::difference_type \_\_parallel\_sort\_qs\_divide (_RAIter __begin, _RAIter __end, _Compare __comp, typename std::iterator_traits<_RAIter>::difference_type __pivot_rank, typename std::iterator_traits<_RAIter>::difference_type __num_samples, \_ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare> void \_\_parallel\_sort\_qsb (_RAIter __begin, _RAIter __end, _Compare __comp, \_ThreadIndex __num_threads)`
- `template<typename _Iter, class _OutputIterator, class _BinaryPredicate> _OutputIterator \_\_parallel\_unique\_copy (_Iter __first, _Iter __last, _OutputIterator __result, _BinaryPredicate __binary_pred)`
- `template<typename _Iter, class _OutputIterator> _OutputIterator \_\_parallel\_unique\_copy (_Iter __first, _Iter __last, _OutputIterator __result)`
- `template<typename _RAIter, typename _Compare> void \_\_qsb\_conquer (_QSBThreadLocal<_RAIter> **__tls, _RAIter __begin, _RAIter __end, _Compare __comp, \_ThreadIndex __iam, \_ThreadIndex __num_threads, bool __parent_wait)`
- `template<typename _RAIter, typename _Compare> std::iterator_traits<_RAIter>::difference_type \_\_qsb\_divide (_RAIter __begin, _RAIter __end, _Compare __comp, \_ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare> void \_\_qsb\_local\_sort\_with\_helping (_QSBThreadLocal<_RAIter> **__tls, _Compare &__comp, \_ThreadIndex __iam, bool __wait)`
- `template<typename _RandomNumberGenerator> int \_\_random\_number\_pow2 (int __logp, _RandomNumberGenerator &__rng)`
- `template<typename _Size> _Size \_\_rd\_log2 (_Size __n)`
- `template<typename _Tp> _Tp \_\_round\_up\_to\_pow2 (_Tp __x)`

- `template<typename __RAIter1, typename __RAIter2, typename _Pred> __RAIter1 \_\_search\_template (__RAIter1 __begin1, __RAIter1 __end1, __RAIter2 __begin2, __RAIter2 __end2, _Pred __pred)`
- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare> _RAIter3 \_\_sequential\_multiway\_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type>::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _RAIter, typename _RandomNumberGenerator> void \_\_sequential\_random\_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator & __rng)`
- `template<typename _Iter> void \_\_shrink (std::vector< _Iter > & __os_starts, size_t & __count_to_two, size_t & __range_length)`
- `template<typename _Iter> void \_\_shrink\_and\_double (std::vector< _Iter > & __os_starts, size_t & __count_to_two, size_t & __range_length, const bool __make_twice)`
- `void \_\_yield ()`
- `template<typename _Iter, typename _FunctorType> size_t list\_partition (const _Iter __begin, const _Iter __end, _Iter * __starts, size_t * __lengths, const int __num_parts, _FunctorType & __f, int __oversampling=0)`
- `template<typename _Tp> const _Tp & max (const _Tp & __a, const _Tp & __b)`
- `template<typename _Tp> const _Tp & min (const _Tp & __a, const _Tp & __b)`
- `template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename _Compare> void multiseq\_partition (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankIterator __begin_offsets, _Compare __comp=std::less< typename std::iterator_traits< typename std::iterator_traits< _RanSeqs >::value_type>::first_type >::value_type >())`
- `template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare> _Tp multiseq\_selection (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankType & __offset, _Compare __comp=std::less< _Tp >())`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu\_parallel::sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<template< typename RAIter, typename C> class iterator, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare> _RAIter3 multiway\_merge\_3\_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<template< typename RAIter, typename C> class iterator, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare> _RAIter3 multiway\_merge\_4\_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType> void multiway\_merge\_exact\_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > * __pieces)`
- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare> _RAIter3 multiway\_merge\_loser\_tree (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`



- `template<typename UnguardedLoserTree, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare > _RAIter3 multiway\_merge\_loser\_tree\_sentinel (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type>::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare > _RAIter3 multiway\_merge\_loser\_tree\_unguarded (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type>::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType > void multiway\_merge\_sampling\_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > * __pieces)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare > _RAIterOut multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu_parallel::sequential_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare > _RAIterOut multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu_parallel::exact_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare > _RAIterOut multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare > _RAIterOut multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare > _RAIterOut multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)`
- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Splitter, typename _Compare > _RAIter3 parallel\_multiway\_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _Splitter __splitter, _DifferenceTp __length, _Compare __comp, ThreadIndex __num_threads)`
- `template<bool __stable, bool __exact, typename _RAIter, typename _Compare > void parallel\_sort\_mwms (_RAIter __begin, _RAIter __end, _Compare __comp, ThreadIndex __num_threads)`
- `template<bool __stable, bool __exact, typename _RAIter, typename _Compare > void parallel\_sort\_mwms\_pu (PMWMS_SortingData< _RAIter > * __sd, _Compare & __comp)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare > _RAIterOut stable\_multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu_parallel::sequential_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare > _RAIterOut stable\_multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu_parallel::exact_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare > _RAIterOut stable\_multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare > _RAIterOut stable\_multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare > _RAIterOut stable\_multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare > _RAIterOut stable\_multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu_parallel::sequential_tag)`



- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`

## Variables

- `static const int \_CASable\_bits`
- `static const \_CASable \_CASable\_mask`

### 3.6.1 Detailed Description

GNU parallel code for public use.

### 3.6.2 Typedef Documentation

#### 3.6.2.1 `typedef unsigned short \_\_gnu\_parallel::\_BinIndex`

Type to hold the index of a bin.

Since many variables of this type are allocated, it should be chosen as small as possible.

Definition at line 47 of file `random_shuffle.h`.

#### 3.6.2.2 `typedef int64_t \_\_gnu\_parallel::\_CASable`

Longest compare-and-swappable integer type on this platform.

Definition at line 127 of file `types.h`.

#### 3.6.2.3 `typedef uint64_t \_\_gnu\_parallel::\_SequenceIndex`

Unsigned integer to index `__elements`. The total number of elements for each algorithm must fit into this type.

Definition at line 117 of file `types.h`.

#### 3.6.2.4 `typedef uint16_t \_\_gnu\_parallel::\_ThreadIndex`

Unsigned integer to index a thread number. The maximum thread number (for each processor) must fit into this type.

Definition at line 123 of file `types.h`.

### 3.6.3 Enumeration Type Documentation

#### 3.6.3.1 `enum \_\_gnu\_parallel::\_AlgorithmStrategy`

Strategies for run-time algorithm selection:

Definition at line 67 of file types.h.

### 3.6.3.2 enum \_\_gnu\_parallel::\_FindAlgorithm

Find algorithms:

Definition at line 106 of file types.h.

### 3.6.3.3 enum \_\_gnu\_parallel::\_MultiwayMergeAlgorithm

Merging algorithms:

Definition at line 85 of file types.h.

### 3.6.3.4 enum \_\_gnu\_parallel::\_Parallelism

Run-time equivalents for the compile-time tags.

Enumerator

***sequential*** Not parallel.

***parallel\_unbalanced*** Parallel unbalanced (equal-sized chunks).

***parallel\_balanced*** Parallel balanced (work-stealing).

***parallel\_omp\_loop*** Parallel with OpenMP dynamic load-balancing.

***parallel\_omp\_loop\_static*** Parallel with OpenMP static load-balancing.

***parallel\_taskqueue*** Parallel with OpenMP taskqueue construct.

Definition at line 44 of file types.h.

### 3.6.3.5 enum \_\_gnu\_parallel::\_PartialSumAlgorithm

Partial sum algorithms: recursive, linear.

Definition at line 91 of file types.h.

### 3.6.3.6 enum \_\_gnu\_parallel::\_SortAlgorithm

Sorting algorithms:

Definition at line 76 of file types.h.

### 3.6.3.7 enum \_\_gnu\_parallel::\_SplittingAlgorithm

Sorting/merging algorithms: sampling, \_\_exact.

Definition at line 98 of file types.h.

## 3.6.4 Function Documentation

3.6.4.1 `template<typename _RAIter, typename _DifferenceTp> void __gnu_parallel::_calc_borders ( _RAIter __elements, _DifferenceTp __length, _DifferenceTp * __off )`

Precalculate \_\_advances for Knuth-Morris-Pratt algorithm.

## Parameters

<code>__elements</code>	Begin iterator of sequence to search for.
<code>__length</code>	Length of sequence to search for.
<code>__off</code>	Returned <code>__offsets</code> .

Definition at line 51 of file `search.h`.

Referenced by `__search_template()`.

**3.6.4.2** `template<typename _Tp> bool __gnu_parallel::__compare_and_swap ( volatile _Tp * __ptr, _Tp __comparand, _Tp __replacement ) [inline]`

Compare-and-swap.

Compare `*__ptr` and `__comparand`. If equal, let `*__ptr=__replacement` and return true, return false otherwise.

## Parameters

<code>__ptr</code>	Pointer to signed integer.
<code>__comparand</code>	Compare value.
<code>__replacement</code>	Replacement value.

Definition at line 108 of file `parallel/compatibility.h`.

Referenced by `__parallel_partition()`, `__gnu_parallel::RestrictedBoundedConcurrentQueue< pair< _RAIter, _RAIter > >::pop_back()`, and `__gnu_parallel::RestrictedBoundedConcurrentQueue< pair< _RAIter, _RAIter > >::pop_↵front()`.

**3.6.4.3** `void __gnu_parallel::__decode2 ( _CASable __x, int & __a, int & __b ) [inline]`

Decode two integers from one `gnu_parallel::CASable`.

## Parameters

<code>__x</code>	<code>__gnu_parallel::CASable</code> to decode integers from.
<code>__a</code>	First integer, to be decoded from the most-significant <code>_CASable_bits/2</code> bits of <code>__x</code> .
<code>__b</code>	Second integer, to be encoded in the least-significant <code>_CASable_bits/2</code> bits of <code>__x</code> .

## See also

`__encode2`

Definition at line 133 of file `parallel/base.h`.

References `_CASable_bits`, and `_CASable_mask`.

Referenced by `__gnu_parallel::RestrictedBoundedConcurrentQueue< pair< _RAIter, _RAIter > >::pop_back()`, `__↵gnu_parallel::RestrictedBoundedConcurrentQueue< pair< _RAIter, _RAIter > >::pop_front()`, and `__gnu_parallel::↵_RestrictedBoundedConcurrentQueue< pair< _RAIter, _RAIter > >::push_front()`.

**3.6.4.4** `template<typename _RAIter, typename _DifferenceTp> void __gnu_parallel::__determine_samples ( _PMWSSortingData< _RAIter > * __sd, _DifferenceTp __num_samples )`

Select `_M_samples` from a sequence.

## Parameters

<code>__sd</code>	Pointer to algorithm data. <code>_Result</code> will be placed in <code>__sd-&gt;_M_samples</code> .
<code>__num_samples</code>	Number of <code>_M_samples</code> to select.

Definition at line 97 of file `multiway_mergesort.h`.

References `__equally_split()`, `__gnu_parallel::PMWMSortingData<_RAIter>::_M_samples`, `__gnu_parallel::PMWMSortingData<_RAIter>::_M_source`, and `__gnu_parallel::PMWMSortingData<_RAIter>::_M_starts`.

#### 3.6.4.5 `__CASable __gnu_parallel::encode2 ( int __a, int __b ) [inline]`

Encode two integers into one `gnu_parallel::_CASable`.

## Parameters

<code>__a</code>	First integer, to be encoded in the most-significant <code>_CASable_bits/2</code> bits.
<code>__b</code>	Second integer, to be encoded in the least-significant <code>_CASable_bits/2</code> bits.

## Returns

value encoding `__a` and `__b`.

## See also

`__decode2`

Definition at line 119 of file `parallel/base.h`.

References `_CASable_bits`.

Referenced by `__gnu_parallel::RestrictedBoundedConcurrentQueue< pair<_RAIter, _RAIter> >::_RestrictedBoundedConcurrentQueue()`, `__gnu_parallel::RestrictedBoundedConcurrentQueue< pair<_RAIter, _RAIter> >::pop_back()`, `__gnu_parallel::RestrictedBoundedConcurrentQueue< pair<_RAIter, _RAIter> >::pop_front()`, and `__gnu_parallel::RestrictedBoundedConcurrentQueue< pair<_RAIter, _RAIter> >::push_front()`.

#### 3.6.4.6 `template<typename _DifferenceType, typename _OutputIterator> _OutputIterator __gnu_parallel::equally_split ( _DifferenceType __n, _ThreadIndex __num_threads, _OutputIterator __s )`

function to split a sequence into parts of almost equal size.

The resulting sequence `__s` of length `__num_threads+1` contains the splitting positions when splitting the range `[0, __n)` into parts of almost equal size (plus minus 1). The first entry is 0, the last one `n`. There may result empty parts.

## Parameters

<code>__n</code>	Number of elements
<code>__num_threads</code>	Number of parts
<code>__s</code>	Splitters

## Returns

End of `__splitter` sequence, i.e. `__s+__num_threads+1`

Definition at line 48 of file `equally_split.h`.

Referenced by `__determine_samples()`, `__find_template()`, `__parallel_partial_sum_linear()`, `__parallel_unique_copy()`, `__search_template()`, and `multiway_merge_exact_splitting()`.

3.6.4.7 `template<typename _DifferenceType > _DifferenceType __gnu_parallel::__equally_split_point ( _DifferenceType __n, _ThreadIndex __num_threads, _ThreadIndex __thread_no )`

function to split a sequence into parts of almost equal size.

Returns the position of the splitting point between thread number `__thread_no` (included) and thread number `__thread_no+1` (excluded).

#### Parameters

<code>__n</code>	Number of elements
<code>__num_threads</code>	Number of parts
<code>__thread_no</code>	Number of threads

#### Returns

splitting point

Definition at line 75 of file `equally_split.h`.

Referenced by `__for_each_template_random_access_ed()`.

3.6.4.8 `template<typename _Tp > _Tp __gnu_parallel::__fetch_and_add ( volatile _Tp * __ptr, _Tp __addend ) [inline]`

Add a value to a variable, atomically.

#### Parameters

<code>__ptr</code>	Pointer to a signed integer.
<code>__addend</code>	Value to add.

Definition at line 74 of file `parallel/compatibility.h`.

Referenced by `__parallel_partition()`, and `__gnu_parallel::__RestrictedBoundedConcurrentQueue< pair< _RAlter, _RAlter > >::push_front()`.

3.6.4.9 `template<typename _RAlter1, typename _RAlter2, typename _Pred, typename _Selector > std::pair< _RAlter1, _RAlter2> __gnu_parallel::__find_template ( _RAlter1 __begin1, _RAlter1 __end1, _RAlter2 __begin2, _Pred __pred, _Selector __selector ) [inline]`

Parallel `std::find`, switch for different algorithms.

#### Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence. Must have same length as first sequence.
<code>__pred</code>	Find predicate.
<code>__selector</code>	_Functionality (e. g. <code>std::find_if()</code> , <code>std::equal()</code> ,...)

#### Returns

Place of finding in both sequences.

Definition at line 60 of file `find.h`.

References `__gnu_parallel::__Settings::get()`, and `std::make_pair()`.

3.6.4.10 `template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector > std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_template ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, equal_split_tag )`

Parallel `std::find`, equal splitting variant.

## Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence. Second <code>__sequence</code> must have same length as first sequence.
<code>__pred</code>	Find predicate.
<code>__selector</code>	<code>__Functionality</code> (e. g. <code>std::find_if()</code> , <code>std::equal()</code> ,...)

## Returns

Place of finding in both sequences.

Definition at line 97 of file `find.h`.

References `__equally_split()`, and `_GLIBCXX_CALL`.

3.6.4.11 `template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector > std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_template ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, growing_blocks_tag )`

Parallel `std::find`, growing block size variant.

## Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence. Second <code>__sequence</code> must have same length as first sequence.
<code>__pred</code>	Find predicate.
<code>__selector</code>	<code>__Functionality</code> (e. g. <code>std::find_if()</code> , <code>std::equal()</code> ,...)

## Returns

Place of finding in both sequences.

## See also

`__gnu_parallel::_Settings::find_sequential_search_size`  
`__gnu_parallel::_Settings::find_scale_factor`

There are two main differences between the growing blocks and the constant-size blocks variants. 1. For GB, the block size grows; for CSB, the block size is fixed. 2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

Definition at line 185 of file `find.h`.

References `_GLIBCXX_CALL`, `__gnu_parallel::_Settings::find_scale_factor`, `__gnu_parallel::_Settings::find_↔ sequential_search_size`, and `__gnu_parallel::_Settings::get()`.

3.6.4.12 `template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector > std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_template ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, constant_size_blocks_tag )`

Parallel `std::find`, constant block size variant.

## Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence. Second <code>__sequence</code> must have same length as first sequence.
<code>__pred</code>	Find predicate.
<code>__selector</code>	<code>_Functionality</code> (e. g. <code>std::find_if()</code> , <code>std::equal()</code> ,...)

## Returns

Place of finding in both sequences.

## See also

`__gnu_parallel::Settings::find_sequential_search_size`  
`__gnu_parallel::Settings::find_block_size` There are two main differences between the growing blocks and the constant-size blocks variants. 1. For GB, the block size grows; for CSB, the block size is fixed. 2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

Definition at line 315 of file `find.h`.

References `_GLIBCXX_CALL`, `__gnu_parallel::Settings::find_initial_block_size`, `__gnu_parallel::Settings::find_sequential_search_size`, and `__gnu_parallel::Settings::get()`.

3.6.4.13 `template<typename _Iter , typename _UserOp , typename _Functionality , typename _Red , typename _Result > _UserOp  
__gnu_parallel::for_each_template_random_access ( _Iter __begin, _Iter __end, _UserOp __user_op, _Functionality &  
__functionality, _Red __reduction, _Result __reduction_start, _Result & __output, typename std::iterator_traits< _Iter  
>::difference_type __bound, _Parallelism __parallelism_tag )`

Chose the desired algorithm by evaluating `__parallelism_tag`.

## Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__user_op</code>	A user-specified functor (comparator, predicate, associative operator,...)
<code>__functionality</code>	functor to <i>process</i> an element with <code>__user_op</code> (depends on desired functionality, e. g. <code>accumulate</code> , <code>for_each</code> ,...)
<code>__reduction</code>	Reduction functor.
<code>__reduction_start</code>	Initial value for reduction.
<code>__output</code>	Output iterator.
<code>__bound</code>	Maximum number of elements processed.
<code>__parallelism_tag</code>	Parallelization method

Definition at line 61 of file `for_each.h`.

References `__for_each_template_random_access_ed()`, `__for_each_template_random_access_omp_loop()`, `__for_each_template_random_access_workstealing()`, `parallel_omp_loop`, `parallel_omp_loop_static`, and `parallel_unbalanced`.

3.6.4.14 `template<typename _RAIter , typename _Op , typename _Fu , typename _Red , typename _Result > _Op  
__gnu_parallel::for_each_template_random_access_ed ( _RAIter __begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r,  
_Result __base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type __bound )`

Embarrassingly parallel algorithm for random access iterators, using hand-crafted parallelization by equal splitting the work.



## Parameters

<code>__begin</code>	Begin iterator of element sequence.
<code>__end</code>	End iterator of element sequence.
<code>__o</code>	User-supplied functor (comparator, predicate, adding functor, ...)
<code>__f</code>	Functor to "process" an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...).
<code>__r</code>	Functor to "add" a single <code>__result</code> to the already processed elements (depends on functionality).
<code>__base</code>	Base value for reduction.
<code>__output</code>	Pointer to position where final result is written to
<code>__bound</code>	Maximum number of elements processed (e. g. for <code>std::count_n()</code> ).

## Returns

User-supplied functor (that may contain a part of the result).

Definition at line 67 of file `par_loop.h`.

References `__equally_split_point()`.

Referenced by `__for_each_template_random_access()`.

```
3.6.4.15 template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result> _Op
__gnu_parallel::__for_each_template_random_access_omp_loop( _RAIter __begin, _RAIter __end, _Op __o, _Fu & __f,
_Red __r, _Result __base, _Result & __output, typename std::iterator_traits<_RAIter>::difference_type __bound )
```

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop.

## Parameters

<code>__begin</code>	Begin iterator of element sequence.
<code>__end</code>	End iterator of element sequence.
<code>__o</code>	User-supplied functor (comparator, predicate, adding functor, etc.).
<code>__f</code>	Functor to <i>process</i> an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...).
<code>__r</code>	Functor to <i>add</i> a single <code>__result</code> to the already processed elements (depends on functionality).
<code>__base</code>	Base value for reduction.
<code>__output</code>	Pointer to position where final result is written to
<code>__bound</code>	Maximum number of elements processed (e. g. for <code>std::count_n()</code> ).

## Returns

User-supplied functor (that may contain a part of the result).

Definition at line 67 of file `omp_loop.h`.

Referenced by `__for_each_template_random_access()`.

```
3.6.4.16 template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result> _Op
__gnu_parallel::__for_each_template_random_access_omp_loop_static( _RAIter __begin, _RAIter __end, _Op __o, _Fu &
__f, _Red __r, _Result __base, _Result & __output, typename std::iterator_traits<_RAIter>::difference_type __bound )
```

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop with static scheduling.

## Parameters

<code>__begin</code>	Begin iterator of element sequence.
<code>__end</code>	End iterator of element sequence.
<code>__o</code>	User-supplied functor (comparator, predicate, adding functor, ...).
<code>__f</code>	Functor to <i>process</i> an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...).
<code>__r</code>	Functor to <i>add</i> a single <code>__result</code> to the already processed <code>__elements</code> (depends on functionality).
<code>__base</code>	Base value for reduction.
<code>__output</code>	Pointer to position where final result is written to
<code>__bound</code>	Maximum number of elements processed (e. g. for <code>std::count_n()</code> ).

## Returns

User-supplied functor (that may contain a part of the result).

Definition at line 66 of file `omp_loop_static.h`.

```
3.6.4.17 template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result > _Op
__gnu_parallel::__for_each_template_random_access_workstealing ( _RAIter __begin, _RAIter __end, _Op __op, _Fu &
__f, _Red __r, _Result __base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type __bound )
```

Work stealing algorithm for random access iterators.

Uses  $O(1)$  additional memory. Synchronization at job lists is done with atomic operations.

## Parameters

<code>__begin</code>	Begin iterator of element sequence.
<code>__end</code>	End iterator of element sequence.
<code>__op</code>	User-supplied functor (comparator, predicate, adding functor, ...).
<code>__f</code>	Functor to <i>process</i> an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...).
<code>__r</code>	Functor to <i>add</i> a single <code>__result</code> to the already processed elements (depends on functionality).
<code>__base</code>	Base value for reduction.
<code>__output</code>	Pointer to position where final result is written to
<code>__bound</code>	Maximum number of elements processed (e. g. for <code>std::count_n()</code> ).

## Returns

User-supplied functor (that may contain a part of the result).

Definition at line 99 of file `workstealing.h`.

References `__gnu_debug::__base()`, `__yield()`, `_GLIBCXX_CALL`, `__gnu_parallel::__Job< _DifferenceTp >::__M_first`, `__gnu_parallel::__Job< _DifferenceTp >::__M_last`, `__gnu_parallel::__Job< _DifferenceTp >::__M_load`, `__gnu_parallel::__Settings::cache_line_size`, `__gnu_parallel::__Settings::get()`, and `min()`.

Referenced by `__for_each_template_random_access()`.

```
3.6.4.18 template<typename _Iter, typename _Compare > bool __gnu_parallel::__is_sorted ( _Iter __begin, _Iter __end,
_Compare __comp )
```

Check whether `[__begin, __end)` is sorted according to `__comp`.

**Parameters**

<code>__begin</code>	Begin iterator of sequence.
<code>__end</code>	End iterator of sequence.
<code>__comp</code>	Comparator.

**Returns**

`true` if sorted, `false` otherwise.

Definition at line 51 of file `checkers.h`.

Referenced by `__sequential_multiway_merge()`, `multiway_merge_loser_tree_sentinel()`, and `parallel_multiway_merge()`.

**3.6.4.19** `template<typename _RAIter, typename _Compare> _RAIter __gnu_parallel::__median_of_three_iterators ( _RAIter __a, _RAIter __b, _RAIter __c, _Compare __comp )`

Compute the median of three referenced elements, according to `__comp`.

**Parameters**

<code>__a</code>	First iterator.
<code>__b</code>	Second iterator.
<code>__c</code>	Third iterator.
<code>__comp</code>	Comparator.

Definition at line 398 of file `parallel/base.h`.

Referenced by `__qsb_divide()`.

**3.6.4.20** `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare> _OutputIterator __gnu_parallel::__merge_advance ( _RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp ) [inline]`

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. Static switch on whether to use the conditional-move variant.

**Parameters**

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__end2</code>	End iterator of second sequence.
<code>__target</code>	Target begin iterator.
<code>__max_length</code>	Maximum number of elements to merge.
<code>__comp</code>	Comparator.

**Returns**

Output end iterator.

Definition at line 171 of file `merge.h`.

References `__merge_advance_movc()`, and `_GLIBCXX_CALL`.

Referenced by `__parallel_merge_advance()`, and `__sequential_multiway_merge()`.

3.6.4.21 `template<typename _RAIter1 , typename _RAIter2 , typename _OutputIterator , typename _DifferenceTp , typename _Compare > _OutputIterator __gnu_parallel::__merge_advance_movc ( _RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp )`

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. Specially designed code should allow the compiler to generate conditional moves instead of branches.

#### Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__end2</code>	End iterator of second sequence.
<code>__target</code>	Target begin iterator.
<code>__max_length</code>	Maximum number of elements to merge.
<code>__comp</code>	Comparator.

#### Returns

Output end iterator.

Definition at line 105 of file `merge.h`.

Referenced by `__merge_advance()`.

3.6.4.22 `template<typename _RAIter1 , typename _RAIter2 , typename _OutputIterator , typename _DifferenceTp , typename _Compare > _OutputIterator __gnu_parallel::__merge_advance_usual ( _RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp )`

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant.

#### Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__end2</code>	End iterator of second sequence.
<code>__target</code>	Target begin iterator.
<code>__max_length</code>	Maximum number of elements to merge.
<code>__comp</code>	Comparator.

#### Returns

Output end iterator.

Definition at line 57 of file `merge.h`.

3.6.4.23 `template<typename _RAIter1 , typename _RAIter2 , typename _RAIter3 , typename _Compare > _RAIter3 __gnu_parallel::__parallel_merge_advance ( _RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2 __end2, _RAIter3 __target, typename std::iterator_traits< _RAIter1 >::difference_type __max_length, _Compare __comp ) [inline]`

Merge routine fallback to sequential in case the iterators of the two input sequences are of different type.

## Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__end2</code>	End iterator of second sequence.
<code>__target</code>	Target begin iterator.
<code>__max_length</code>	Maximum number of elements to merge.
<code>__comp</code>	Comparator.

## Returns

Output end iterator.

Definition at line 195 of file merge.h.

References `__merge_advance()`.

```
3.6.4.24  template<typename _RAIter1, typename _RAIter3, typename _Compare > _RAIter3 __gnu_parallel::__parallel_merge_↵
advance ( _RAIter1 & __begin1, _RAIter1 __end1, _RAIter1 & __begin2, _RAIter1 __end2, _RAIter3 __target, typename
std::iterator_traits< _RAIter1 >::difference_type __max_length, _Compare __comp ) [inline]
```

Parallel merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. The functionality is projected onto `parallel_multiway_merge`.

## Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__end2</code>	End iterator of second sequence.
<code>__target</code>	Target begin iterator.
<code>__max_length</code>	Maximum number of elements to merge.
<code>__comp</code>	Comparator.

## Returns

Output end iterator.

Definition at line 223 of file merge.h.

References `std::make_pair()`, `multiway_merge_exact_splitting()`, and `parallel_multiway_merge()`.

```
3.6.4.25  template<typename _RAIter, typename _Compare > void __gnu_parallel::__parallel_nth_element ( _RAIter __begin,
_RAlter __nth, _RAIter __end, _Compare __comp )
```

Parallel implementation of `std::nth_element()`.

## Parameters

<code>__begin</code>	Begin iterator of input sequence.
----------------------	-----------------------------------

<code>__nth</code>	Iterator of element that must be in position afterwards.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

Definition at line 332 of file partition.h.

References `__parallel_partition()`, `_GLIBCXX_CALL`, `__gnu_parallel::_Settings::get()`, `std::max()`, `__gnu_parallel::_Settings::nth_element_minimal_n`, and `__gnu_parallel::_Settings::partition_minimal_n`.

Referenced by `__parallel_partial_sort()`.

**3.6.4.26** `template<typename _RAIter, typename _Compare> void __gnu_parallel::_parallel_partial_sort ( _RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp )`

Parallel implementation of `std::partial_sort()`.

Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__middle</code>	Sort until this position.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

Definition at line 422 of file partition.h.

References `__parallel_nth_element()`.

**3.6.4.27** `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation> _OutputIterator __gnu_parallel::_parallel_partial_sum ( _Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op )`

Parallel partial sum front-\_\_end.

Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__result</code>	Begin iterator of output sequence.
<code>__bin_op</code>	Associative binary function.

Returns

End iterator of output sequence.

Definition at line 205 of file partial\_sum.h.

References `__parallel_partial_sum_linear()`, `_GLIBCXX_CALL`, and `__gnu_parallel::_Settings::get()`.

**3.6.4.28** `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation> _OutputIterator __gnu_parallel::_parallel_partial_sum_basecase ( _Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator_traits<_Iter>::value_type __value )`

Base case prefix sum routine.

Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__result</code>	Begin iterator of output sequence.
<code>__bin_op</code>	Associative binary function.
<code>__value</code>	Start value. Must be passed since the neutral element is unknown in general.

#### Returns

End iterator of output sequence.

Definition at line 58 of file `partial_sum.h`.

Referenced by `__parallel_partial_sum_linear()`.

```
3.6.4.29 template<typename _Iter , typename _OutputIterator , typename _BinaryOperation > _OutputIterator
    __gnu_parallel::__parallel_partial_sum_linear ( _Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation
        __bin_op, typename std::iterator_traits< _Iter >::difference_type __n )
```

Parallel partial sum implementation, two-phase approach, no recursion.

#### Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__result</code>	Begin iterator of output sequence.
<code>__bin_op</code>	Associative binary function.
<code>__n</code>	Length of sequence.

#### Returns

End iterator of output sequence.

Definition at line 89 of file `partial_sum.h`.

References `__equally_split()`, `__parallel_partial_sum_basecase()`, `std::accumulate()`, `__gnu_parallel::_Settings::get()`, and `__gnu_parallel::_Settings::partial_sum_dilation`.

Referenced by `__parallel_partial_sum()`.

```
3.6.4.30 template<typename _RAIter , typename _Predicate > std::iterator_traits<_RAIter>::difference_type
    __gnu_parallel::__parallel_partition ( _RAIter __begin, _RAIter __end, _Predicate __pred, _ThreadIndex __num_threads
    )
```

Parallel implementation of `std::partition`.

#### Parameters

<code>__begin</code>	Begin iterator of input sequence to split.
<code>__end</code>	End iterator of input sequence to split.
<code>__pred</code>	Partition predicate, possibly including some kind of pivot.
<code>__num_threads</code>	Maximum number of threads to use for this task.

#### Returns

Number of elements not fulfilling the predicate.

Definition at line 56 of file `partition.h`.

References `__compare_and_swap()`, `__fetch_and_add()`, `_GLIBCXX_CALL`, `_GLIBCXX_VOLATILE`, `__gnu_parallel::Settings::get()`, `__gnu_parallel::Settings::partition_chunk_share`, and `__gnu_parallel::Settings::partition_chunk_size`.

Referenced by `__parallel_nth_element()`, `__parallel_sort_qs_divide()`, and `__qsb_divide()`.

3.6.4.31 `template<typename _RAIter, typename _RandomNumberGenerator > void __gnu_parallel::__parallel_random_shuffle ( _RAIter __begin, _RAIter __end, _RandomNumberGenerator __rng = _RandomNumber() ) [inline]`

Parallel random public call.

Parameters

<code>__begin</code>	Begin iterator of sequence.
<code>__end</code>	End iterator of sequence.
<code>__rng</code>	Random number generator to use.

Definition at line 522 of file `random_shuffle.h`.

References `__parallel_random_shuffle_drs()`.

3.6.4.32 `template<typename _RAIter, typename _RandomNumberGenerator > void __gnu_parallel::__parallel_random_shuffle_drs ( _RAIter __begin, _RAIter __end, typename std::iterator_traits<_RAIter>::difference_type __n, _ThreadIndex __num_threads, _RandomNumberGenerator & __rng )`

Main parallel random shuffle step.

Parameters

<code>__begin</code>	Begin iterator of sequence.
<code>__end</code>	End iterator of sequence.
<code>__n</code>	Length of sequence.
<code>__num_threads</code>	Number of threads to use.
<code>__rng</code>	Random number generator to use.

Definition at line 265 of file `random_shuffle.h`.

References `__gnu_parallel::DRSSorterPU<_RAIter, _RandomNumberGenerator>::__bins_end`, `__parallel_random_shuffle_drs_pu()`, `__rd_log2()`, `__round_up_to_pow2()`, `__sequential_random_shuffle()`, `_GLIBCXX_CALL`, `__gnu_parallel::DRandomShufflingGlobalData<_RAIter>::__M_bin_proc`, `__gnu_parallel::DRSSorterPU<_RAIter, _RandomNumberGenerator>::__M_bins_begin`, `__gnu_parallel::DRandomShufflingGlobalData<_RAIter>::__M_dist`, `__gnu_parallel::DRandomShufflingGlobalData<_RAIter>::__M_num_bins`, `__gnu_parallel::DRandomShufflingGlobalData<_RAIter>::__M_num_bits`, `__gnu_parallel::DRSSorterPU<_RAIter, _RandomNumberGenerator>::__M_num_threads`, `__gnu_parallel::DRSSorterPU<_RAIter, _RandomNumberGenerator>::__M_sd`, `__gnu_parallel::DRSSorterPU<_RAIter, _RandomNumberGenerator>::__M_seed`, `__gnu_parallel::DRandomShufflingGlobalData<_RAIter>::__M_starts`, `__gnu_parallel::DRandomShufflingGlobalData<_RAIter>::__M_temporaries`, `__gnu_parallel::Settings::get()`, `__gnu_parallel::Settings::L2_cache_size`, `std::max()`, `std::min()`, and `__gnu_parallel::Settings::TLB_size`.

Referenced by `__parallel_random_shuffle()`.

3.6.4.33 `template<typename _RAIter, typename _RandomNumberGenerator > void __gnu_parallel::__parallel_random_shuffle_drs_pu ( _DRSSorterPU<_RAIter, _RandomNumberGenerator> * __pus )`

Random shuffle code executed by each thread.



## Parameters

<code>__pus</code>	Array of thread-local data records.
--------------------	-------------------------------------

Definition at line 122 of file `random_shuffle.h`.

References `__random_number_pow2()`, `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_dist`, `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_num_bins`, `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_num_bits`, `__gnu_parallel::_DRSSorterPU<_RAIter,_RandomNumberGenerator>::_M_num_threads`, `__gnu_parallel::_DRSSorterPU<_RAIter,_RandomNumberGenerator>::_M_sd`, `__gnu_parallel::_DRSSorterPU<_RAIter,_RandomNumberGenerator>::_M_seed`, `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_starts`, and `std::partial_sum()`.

Referenced by `__parallel_random_shuffle_drs()`.

3.6.4.34 `template<bool __stable, typename _RAIter, typename _Compare> void __gnu_parallel::_parallel_sort ( _RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_tag __parallelism ) [inline]`

Choose multiway mergesort, splitting variant at run-time, for parallel sorting.

## Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

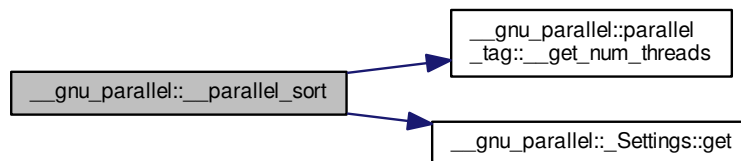
## Template Parameters

<code>__stable</code>	Sort stable.
-----------------------	--------------

Definition at line 75 of file `sort.h`.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `_GLIBCXX_CALL`, and `__gnu_parallel::_Settings::__get()`.

Here is the call graph for this function:



3.6.4.35 `template<bool __stable, typename _RAIter, typename _Compare> void __gnu_parallel::_parallel_sort ( _RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_exact_tag __parallelism ) [inline]`

Choose multiway mergesort with exact splitting, for parallel sorting.

## Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

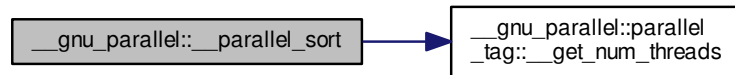
## Template Parameters

<code>__stable</code>	Sort stable.
-----------------------	--------------

Definition at line 99 of file `sort.h`.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



3.6.4.36 `template<bool __stable, typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_sampling_tag __parallelism ) [inline]`

Choose multiway mergesort with splitting by sampling, for parallel sorting.

## Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

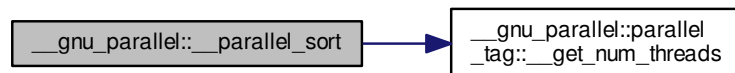
## Template Parameters

<code>__stable</code>	Sort stable.
-----------------------	--------------

Definition at line 120 of file `sort.h`.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



3.6.4.37 `template<bool __stable, typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter __end, _Compare __comp, quicksort_tag __parallelism ) [inline]`

Choose quicksort for parallel sorting.

## Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

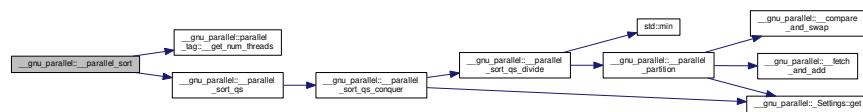
## Template Parameters

<code>__stable</code>	Sort stable.
-----------------------	--------------

Definition at line 140 of file `sort.h`.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `__parallel_sort_qs()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



3.6.4.38 `template<bool __stable, typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter __end, _Compare __comp, balanced_quicksort_tag __parallelism ) [inline]`

Choose balanced quicksort for parallel sorting.

## Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

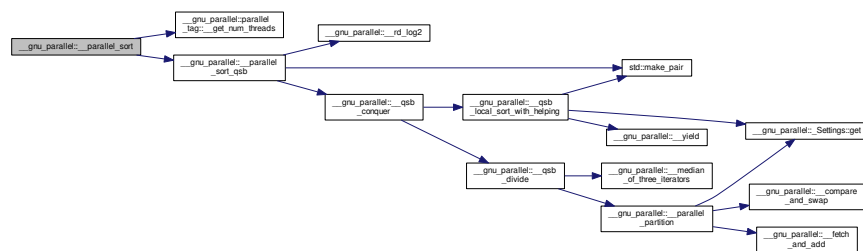
## Template Parameters

<code>__stable</code>	Sort stable.
-----------------------	--------------

Definition at line 161 of file `sort.h`.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `__parallel_sort_qsb()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



3.6.4.39 `template<bool __stable, typename _RAIter, typename _Compare > void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter __end, _Compare __comp, default_parallel_tag __parallelism ) [inline]`

Choose multiway mergesort with exact splitting, for parallel sorting.

## Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

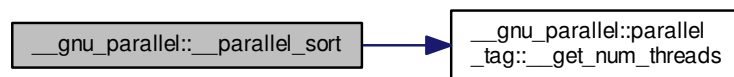
## Template Parameters

<code>__stable</code>	Sort stable.
-----------------------	--------------

Definition at line 183 of file `sort.h`.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



3.6.4.40 `template<bool __stable, typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter __end, _Compare __comp, parallel_tag __parallelism ) [inline]`

Choose a parallel sorting algorithm.

## Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

## Template Parameters

<code>__stable</code>	Sort stable.
-----------------------	--------------

Definition at line 203 of file `sort.h`.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `__parallel_sort_qs()`, `__parallel_sort_qsb()`, `_GLIBCXX_CALL`, and `__gnu_parallel::_Settings::get()`.



## Parameters

<code>__begin</code>	Begin iterator of subsequence.
<code>__end</code>	End iterator of subsequence.
<code>__comp</code>	Comparator.
<code>__pivot_rank</code>	Desired <code>__rank</code> of the pivot.
<code>__num_samples</code>	Choose pivot from that many samples.
<code>__num_threads</code>	Number of threads that are allowed to work on this part.

Definition at line 51 of file quicksort.h.

References `__parallel_partition()`, and `std::min()`.

Referenced by `__parallel_sort_qs_conquer()`.

3.6.4.44 `template<typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_sort_qsb ( _RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads )`

Top-level quicksort routine.

## Parameters

<code>__begin</code>	Begin iterator of sequence.
<code>__end</code>	End iterator of sequence.
<code>__comp</code>	Comparator.
<code>__num_threads</code>	Number of threads that are allowed to work on this part.

Definition at line 430 of file balanced\_quicksort.h.

References `__qsb_conquer()`, `__rd_log2()`, `_GLIBCXX_CALL`, `__gnu_parallel::__QSBThreadLocal<_RAIter>::__M_↔elements_leftover`, and `std::make_pair()`.

Referenced by `__parallel_sort()`.

3.6.4.45 `template<typename _Iter, class _OutputIterator, class _BinaryPredicate> _OutputIterator __gnu_parallel::__parallel_unique_copy ( _Iter __first, _Iter __last, _OutputIterator __result, _BinaryPredicate __binary_pred )`

Parallel `std::unique_copy()`, w/\_o explicit equality predicate.

## Parameters

<code>__first</code>	Begin iterator of input sequence.
<code>__last</code>	End iterator of input sequence.
<code>__result</code>	Begin iterator of result <code>__sequence</code> .
<code>__binary_pred</code>	Equality predicate.

## Returns

End iterator of result `__sequence`.

Definition at line 50 of file unique\_copy.h.

References `__equally_split()`, and `_GLIBCXX_CALL`.

Referenced by `__parallel_unique_copy()`.

3.6.4.46 `template<typename _Iter, class _OutputIterator> _OutputIterator __gnu_parallel::__parallel_unique_copy ( _Iter __first, _Iter __last, _OutputIterator __result ) [inline]`

Parallel `std::unique_copy()`, without explicit equality predicate.



## Parameters

<code>__first</code>	Begin iterator of input sequence.
<code>__last</code>	End iterator of input sequence.
<code>__result</code>	Begin iterator of result <code>__sequence</code> .

## Returns

End iterator of result `__sequence`.

Definition at line 186 of file `unique_copy.h`.

References `__parallel_unique_copy()`.

```
3.6.4.47 template<typename _RAIter, typename _Compare > void __gnu_parallel::__qsb_conquer ( _QSBThreadLocal<
    _RAIter > ** __tls, _RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __iam, _ThreadIndex
    __num_threads, bool __parent_wait )
```

Quicksort conquer step.

## Parameters

<code>__tls</code>	Array of thread-local storages.
<code>__begin</code>	Begin iterator of subsequence.
<code>__end</code>	End iterator of subsequence.
<code>__comp</code>	Comparator.
<code>__iam</code>	Number of the thread processing this function.
<code>__num_threads</code>	Number of threads that are allowed to work on this part.

Definition at line 171 of file `balanced_quicksort.h`.

References `__qsb_divide()`, `__qsb_local_sort_with_helping()`, `__gnu_parallel::__QSBThreadLocal< _RAIter >::__M_←`  
`elements_leftover`, and `__gnu_parallel::__QSBThreadLocal< _RAIter >::__M_initial`.

Referenced by `__parallel_sort_qsb()`.

```
3.6.4.48 template<typename _RAIter, typename _Compare > std::iterator_traits<_RAIter>::difference_type
    __gnu_parallel::__qsb_divide ( _RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads )
```

Balanced quicksort divide step.

## Parameters

<code>__begin</code>	Begin iterator of subsequence.
<code>__end</code>	End iterator of subsequence.
<code>__comp</code>	Comparator.
<code>__num_threads</code>	Number of threads that are allowed to work on this part.

## Precondition

`(__end-__begin)>=1`

Definition at line 100 of file `balanced_quicksort.h`.

References `__median_of_three_iterators()`, and `__parallel_partition()`.

Referenced by `__qsb_conquer()`.

3.6.4.49 `template<typename _RAIter, typename _Compare > void __gnu_parallel::__qsb_local_sort_with_helping (`  
`_QSBThreadLocal<_RAIter > ** __tls, _Compare & __comp, _ThreadIndex __iam, bool __wait )`

Quicksort step doing load-balanced local sort.

## Parameters

<code>__tls</code>	Array of thread-local storages.
<code>__comp</code>	Comparator.
<code>__iam</code>	Number of the thread processing this function.

Definition at line 247 of file `balanced_quicksort.h`.

References `__yield()`, `_GLIBCXX_ASSERTIONS`, `__gnu_parallel::QSBThreadLocal<_RAIter>::M_elements_`, `__gnu_parallel::QSBThreadLocal<_RAIter>::M_initial`, `__gnu_parallel::QSBThreadLocal<_RAIter>::M_leftover_parts`, `__gnu_parallel::QSBThreadLocal<_RAIter>::M_num_threads`, `__gnu_parallel::Settings::get()`, `std::make_pair()`, and `__gnu_parallel::Settings::sort_qsb_base_case_maximal_n`.

Referenced by `__qsb_conquer()`.

**3.6.4.50** `template<typename _RandomNumberGenerator> int __gnu_parallel::__random_number_pow2 ( int __logp, _RandomNumberGenerator & __rng ) [inline]`

Generate a random number in  $[0, 2^{\text{__logp}})$ .

## Parameters

<code>__logp</code>	Logarithm (basis 2) of the upper range <code>__bound</code> .
<code>__rng</code>	Random number generator to use.

Definition at line 115 of file `random_shuffle.h`.

Referenced by `__parallel_random_shuffle_drs_pu()`, and `__sequential_random_shuffle()`.

**3.6.4.51** `template<typename _Size> _Size __gnu_parallel::__rd_log2 ( _Size __n ) [inline]`

Calculates the rounded-down logarithm of `__n` for base 2.

## Parameters

<code>__n</code>	Argument.
------------------	-----------

## Returns

Returns 0 for any argument  $< 1$ .

Definition at line 102 of file `parallel/base.h`.

Referenced by `__parallel_random_shuffle_drs()`, `__parallel_sort_qsb()`, `__round_up_to_pow2()`, `__sequential_random_shuffle()`, `__gnu_parallel::LoserTreeBase<_Tp, _Compare>::LoserTreeBase()`, `multiseq_partition()`, and `multiseq_selection()`.

**3.6.4.52** `template<typename _Tp> _Tp __gnu_parallel::__round_up_to_pow2 ( _Tp __x )`

Round up to the next greater power of 2.

## Parameters

<code>__x</code>	Integer to round up
------------------	---------------------

Definition at line 248 of file `random_shuffle.h`.

References `__rd_log2()`.

Referenced by `__parallel_random_shuffle_drs()`, `__sequential_random_shuffle()`, and `multiseq_selection()`.

3.6.4.53 `template<typename __RAIter1, typename __RAIter2, typename _Pred > __RAIter1 __gnu_parallel::__search_template (`  
`__RAIter1 __begin1, __RAIter1 __end1, __RAIter2 __begin2, __RAIter2 __end2, _Pred __pred )`

Parallel `std::search`.

## Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__end2</code>	End iterator of second sequence.
<code>__pred</code>	Find predicate.

## Returns

Place of finding in first sequences.

Definition at line 81 of file `search.h`.

References `__calc_borders()`, `__equally_split()`, `_GLIBCXX_CALL`, and `std::min()`.

```
3.6.4.54 template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp,
typename _Compare > _RAIter3 __gnu_parallel::__sequential_multiway_merge ( _RAIterIterator __seqs_begin,
_RAlterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits<
_RAlterIterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp )
```

Sequential multi-way merging switch.

The `_GLIBCXX_PARALLEL_DECISION` is based on the branching factor and runtime settings.

## Parameters

<code>__seqs_begin</code>	Begin iterator of iterator pair input sequence.
<code>__seqs_end</code>	End iterator of iterator pair input sequence.
<code>__target</code>	Begin iterator of output sequence.
<code>__comp</code>	Comparator.
<code>__length</code>	Maximum length to merge, possibly larger than the number of elements available.
<code>__sentinel</code>	The sequences have <code>__a</code> <code>__sentinel</code> element.

## Returns

End iterator of output sequence.

Definition at line 920 of file `multiway_merge.h`.

References `__is_sorted()`, `__merge_advance()`, `_GLIBCXX_CALL`, and `_GLIBCXX_PARALLEL_LENGTH`.

Referenced by `multiway_merge()`, and `multiway_merge_sentinels()`.

```
3.6.4.55 template<typename _RAIter, typename _RandomNumberGenerator > void __gnu_parallel::__sequential_random_shuffle (
_RAlter __begin, _RAIter __end, _RandomNumberGenerator & __rng )
```

Sequential cache-efficient random shuffle.

## Parameters

<code>__begin</code>	Begin iterator of sequence.
<code>__end</code>	End iterator of sequence.
<code>__rng</code>	Random number generator to use.

Definition at line 410 of file `random_shuffle.h`.

References `__random_number_pow2()`, `__rd_log2()`, `__round_up_to_pow2()`, `__gnu_parallel::_Settings::get()`, `__gnu_parallel::_Settings::L2_cache_size`, `std::min()`, `std::partial_sum()`, and `__gnu_parallel::_Settings::TLB_size`.

Referenced by `__parallel_random_shuffle_drs()`.

3.6.4.56 `template<typename _Iter > void __gnu_parallel::__shrink ( std::vector< _Iter > & __os_starts, size_t & __count_to_two, size_t & __range_length )`

Combines two ranges into one and thus halves the number of ranges.

Parameters

<code>__os_starts</code>	Start positions worked on (oversampled).
<code>__count_to_two</code>	Counts up to 2.
<code>__range_length</code>	Current length of a chunk.

Definition at line 70 of file `list_partition.h`.

References `std::vector< _Tp, _Alloc >::size()`.

Referenced by `__shrink_and_double()`.

3.6.4.57 `template<typename _Iter > void __gnu_parallel::__shrink_and_double ( std::vector< _Iter > & __os_starts, size_t & __count_to_two, size_t & __range_length, const bool __make_twice )`

Shrinks and doubles the ranges.

Parameters

<code>__os_starts</code>	Start positions worked on (oversampled).
<code>__count_to_two</code>	Counts up to 2.
<code>__range_length</code>	Current length of a chunk.
<code>__make_twice</code>	Whether the <code>__os_starts</code> is allowed to be grown or not

Definition at line 50 of file `list_partition.h`.

References `__shrink()`, `std::vector< _Tp, _Alloc >::resize()`, and `std::vector< _Tp, _Alloc >::size()`.

Referenced by `list_partition()`.

3.6.4.58 `void __gnu_parallel::__yield ( ) [inline]`

Yield control to another thread, without waiting for the end of the time slice.

Definition at line 121 of file `parallel/compatibility.h`.

Referenced by `__for_each_template_random_access_workstealing()`, and `__qsb_local_sort_with_helping()`.

3.6.4.59 `template<typename _Iter, typename _FunctorType > size_t __gnu_parallel::list_partition ( const _Iter __begin, const _Iter __end, _Iter * __starts, size_t * __lengths, const int __num_parts, _FunctorType & __f, int __oversampling = 0 )`

Splits a sequence given by input iterators into parts of almost equal size.

The function needs only one pass over the sequence.

Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__starts</code>	Start iterators for the resulting parts, dimension <code>__num_parts+1</code> . For convenience, <code>__starts [ __num_parts ]</code> contains the end iterator of the sequence.

<code>__lengths</code>	Length of the resulting parts.
<code>__num_parts</code>	Number of parts to split the sequence into.
<code>__f</code>	Functor to be applied to each element by traversing <code>__it</code>
<code>__oversampling</code>	Oversampling factor. If 0, then the partitions will differ in at most $\{ \{ \_end \} - \{ \_begin \} \}$ <code>__</code> elements. Otherwise, the ratio between the longest and the shortest part is bounded by $1 / ( \{ \_oversampling \} \{ num \} )$

### Returns

Length of the whole sequence.

Definition at line 101 of file `list_partition.h`.

References `__shrink_and_double()`, and `std::vector<_Tp, _Alloc>::size()`.

**3.6.4.60** `template<typename _Tp> const _Tp& __gnu_parallel::max ( const _Tp & __a, const _Tp & __b ) [inline]`

Equivalent to `std::max`.

Definition at line 150 of file `parallel/base.h`.

**3.6.4.61** `template<typename _Tp> const _Tp& __gnu_parallel::min ( const _Tp & __a, const _Tp & __b ) [inline]`

Equivalent to `std::min`.

Definition at line 144 of file `parallel/base.h`.

Referenced by `__for_each_template_random_access_workstealing()`.

**3.6.4.62** `template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename  
_Compare> void __gnu_parallel::multiseq_partition ( _RanSeqs __begin_seqs, _RanSeqs  
__end_seqs, _RankType __rank, _RankIterator __begin_offsets, _Compare __comp =  
std::less< typename std::iterator_traits<typename std::iterator<  
_traits<_RanSeqs>::value_type:: first_type>::value_type>() )`

Splits several sorted sequences at a certain global `__rank`, resulting in a splitting point for each sequence. The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty. If there are several equal elements across the split, the ones on the `__left` side will be chosen from sequences with smaller number.

### Parameters

<code>__begin_seqs</code>	Begin of the sequence of iterator pairs.
<code>__end_seqs</code>	End of the sequence of iterator pairs.
<code>__rank</code>	The global rank to partition at.
<code>__begin_offsets</code>	A random-access <code>__sequence</code> <code>__begin</code> where the <code>__result</code> will be stored in. Each element of the sequence is an iterator that points to the first element on the greater part of the respective <code>__sequence</code> .
<code>__comp</code>	The ordering functor, defaults to <code>std::less&lt;_Tp&gt;</code> .

Definition at line 122 of file `multiseq_selection.h`.

References `__rd_log2()`, `_GLIBCXX_CALL`, `std::vector<_Tp, _Alloc>::begin()`, `std::distance()`, `std::priority_queue<_Tp, _Sequence, _Compare>::empty()`, `std::vector<_Tp, _Alloc>::end()`, `std::make_pair()`, `std::max()`, `std::min()`, `std::priority_queue<_Tp, _Sequence, _Compare>::pop()`, `std::priority_queue<_Tp, _Sequence, _Compare>::push()`, `std::vector<_Tp, _Alloc>::push_back()`, and `std::priority_queue<_Tp, _Sequence, _Compare>::top()`.

Referenced by `multiway_merge_exact_splitting()`.

3.6.4.63 `template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare> _Tp  
__gnu_parallel::multiseq_selection ( _RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankType &  
__offset, _Compare __comp = std::less<_Tp>() )`

Selects the element at a certain global `__rank` from several sorted sequences.

The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty.

#### Parameters

<code>__begin_seqs</code>	Begin of the sequence of iterator pairs.
<code>__end_seqs</code>	End of the sequence of iterator pairs.
<code>__rank</code>	The global rank to partition at.
<code>__offset</code>	The rank of the selected element in the global subsequence of elements equal to the selected element. If the selected element is unique, this number is 0.
<code>__comp</code>	The ordering functor, defaults to <code>std::less</code> .

Definition at line 388 of file `multiseq_selection.h`.

References `__rd_log2()`, `__round_up_to_pow2()`, `_GLIBCXX_CALL`, `std::vector<_Tp, _Alloc>::begin()`, `std::vector<_Tp, _Alloc>::distance()`, `std::priority_queue<_Tp, _Sequence, _Compare>::empty()`, `std::vector<_Tp, _Alloc>::end()`, `std::make_pair()`, `std::max()`, `std::min()`, `std::priority_queue<_Tp, _Sequence, _Compare>::pop()`, `std::priority_queue<_Tp, _Sequence, _Compare>::push()`, `std::vector<_Tp, _Alloc>::push_back()`, and `std::priority_queue<_Tp, _Sequence, _Compare>::top()`.

3.6.4.64 `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>  
_RAIterOut __gnu_parallel::multiway_merge ( _RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end,  
_RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag )`

Multiway Merge Frontend.

Merge the sequences specified by `seqs_begin` and `__seqs_end` into `__target`. `__seqs_begin` and `__seqs_end` must point to a sequence of pairs. These pairs must contain an iterator to the beginning of a sequence in their first entry and an iterator the `_M_end` of the same sequence in their second entry.

Ties are broken arbitrarily. See `stable_multiway_merge` for a variant that breaks ties by sequence number but is slower.

The first entries of the pairs (i.e. the begin iterators) will be moved forward.

The output sequence has to provide enough space for all elements that are written to it.

This function will merge the input sequences:

- not stable
- parallel, depending on the input size and Settings
- using sampling for splitting
- not using sentinels

Example:

```
int sequences[10][10];
for (int __i = 0; __i < 10; ++__i)
    for (int __j = 0; __j < 10; ++__j)
        sequences[__i][__j] = __j;

int __out[33];
std::vector<std::pair<int*> > seqs;
```



```

for (int __i = 0; __i < 10; ++__i)
{ seqs.push(std::make_pair<int*>(sequences[__i],
                                sequences[__i] + 10)) }

multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int*>(), 33);

```

**See also**

`stable_multiway_merge`

**Precondition**

All input sequences must be sorted.

Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.

**Postcondition**

[\_\_target, return \_\_value) contains merged \_\_elements from the input sequences.

return \_\_value - \_\_target = min(\_\_length, number of elements in all sequences).

**Template Parameters**

<code>_RAIterPairIterator</code>	iterator over sequence of pairs of iterators
<code>_RAIterOut</code>	iterator over target sequence
<code>_DifferenceTp</code>	difference type for the sequence
<code>_Compare</code>	strict weak ordering type to compare elements in sequences

**Parameters**

<code>__seqs_begin</code>	__begin of sequence __sequence
<code>__seqs_end</code>	__M_end of sequence __sequence
<code>__target</code>	target sequence to merge to.
<code>__comp</code>	strict weak ordering to use for element comparison.
<code>__length</code>	Maximum length to merge, possibly larger than the number of elements available.

**Returns**

\_\_M\_end iterator of output sequence

Definition at line 1418 of file `multiway_merge.h`.

References `__sequential_multiway_merge()`, and `_GLIBCXX_CALL`.

**3.6.4.65** `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare > _RAIter3 __gnu_parallel::multiway_merge_3_variant ( _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp )`

Highly efficient 3-way merging procedure.

Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into labels in C++).

This works well for merging up to 4 sequences.

Note that making the merging stable does *not* come at a performance hit.

Whether the merging is done guarded or unguarded is selected by the used iterator class.

## Parameters

<code>__seqs_begin</code>	Begin iterator of iterator pair input sequence.
<code>__seqs_end</code>	End iterator of iterator pair input sequence.
<code>__target</code>	Begin iterator of output sequence.
<code>__comp</code>	Comparator.
<code>__length</code>	Maximum length to merge, less equal than the total number of elements available.

## Returns

End iterator of output sequence.

Definition at line 241 of file `multiway_merge.h`.

References `_GLIBCXX_CALL`.

**3.6.4.66** `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare > _RAIter3 __gnu_parallel::multiway_merge_4_variant ( _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp )`

Highly efficient 4-way merging procedure.

Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into goto labels in C++).

This works well for merging up to 4 sequences.

Note that making the merging stable does *not* come at a performance hit.

Whether the merging is done guarded or unguarded is selected by the used iterator class.

## Parameters

<code>__seqs_begin</code>	Begin iterator of iterator pair input sequence.
<code>__seqs_end</code>	End iterator of iterator pair input sequence.
<code>__target</code>	Begin iterator of output sequence.
<code>__comp</code>	Comparator.
<code>__length</code>	Maximum length to merge, less equal than the total number of elements available.

## Returns

End iterator of output sequence.

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

References `_GLIBCXX_CALL`.

**3.6.4.67** `template<bool __stable, typename _RAIterIterator , typename _Compare , typename _DifferenceType > void __gnu_parallel::multiway_merge_exact_splitting ( _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > * __pieces )`

Exact splitting for parallel multiway-merge routine.

None of the passed sequences may be empty.

Definition at line 1120 of file `multiway_merge.h`.

References `__equally_split()`, `_GLIBCXX_PARALLEL_LENGTH`, `std::vector< _Tp, _Alloc >::begin()`, `std::vector< _Tp, _Alloc >::end()`, `multiseq_partition()`, and `std::vector< _Tp, _Alloc >::resize()`.

Referenced by `__parallel_merge_advance()`.

```
3.6.4.68 template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >
    _RAIter3 __gnu_parallel::multiway_merge_loser_tree ( _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3
    __target, _DifferenceTp __length, _Compare __comp )
```

Multi-way merging procedure for a high branching factor, guarded case.

This merging variant uses a `LoserTree` class as selected by `_LT`.

Stability is selected through the used `LoserTree` class `_LT`.

At least one non-empty sequence is required.

#### Parameters

<code>__seqs_begin</code>	Begin iterator of iterator pair input sequence.
<code>__seqs_end</code>	End iterator of iterator pair input sequence.
<code>__target</code>	Begin iterator of output sequence.
<code>__comp</code>	Comparator.
<code>__length</code>	Maximum length to merge, less equal than the total number of elements available.

#### Returns

End iterator of output sequence.

Definition at line 491 of file `multiway_merge.h`.

References `_GLIBCXX_CALL`, and `_GLIBCXX_PARALLEL_LENGTH`.

```
3.6.4.69 template<typename UnguardedLoserTree, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp,
    typename _Compare > _RAIter3 __gnu_parallel::multiway_merge_loser_tree_sentinel ( _RAIterIterator __seqs_begin,
    _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits<
    _RAIterIterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp )
```

Multi-way merging procedure for a high branching factor, requiring sentinels to exist.

#### Template Parameters

<i>UnguardedLoserTree</i>	<code>_LoserTree</code> variant to use for the unguarded merging.
---------------------------	---

#### Parameters

<code>__seqs_begin</code>	Begin iterator of iterator pair input sequence.
<code>__seqs_end</code>	End iterator of iterator pair input sequence.
<code>__target</code>	Begin iterator of output sequence.
<code>__comp</code>	Comparator.
<code>__length</code>	Maximum length to merge, less equal than the total number of elements available.

**Returns**

End iterator of output sequence.

Definition at line 662 of file `multiway_merge.h`.

References `__is_sorted()`, and `_GLIBCXX_CALL`.

```
3.6.4.70 template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare
> _RAIter3 __gnu_parallel::multiway_merge_loser_tree_unguarded ( _RAIterIterator __seqs_begin, _RAIterIterator
__seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator
>::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp )
```

Multi-way merging procedure for a high branching factor, unguarded case.

Merging is done using the `LoserTree` class `_LT`.

Stability is selected by the used `LoserTrees`.

**Precondition**

No input will run out of elements during the merge.

**Parameters**

<code>__seqs_begin</code>	Begin iterator of iterator pair input sequence.
<code>__seqs_end</code>	End iterator of iterator pair input sequence.
<code>__target</code>	Begin iterator of output sequence.
<code>__comp</code>	Comparator.
<code>__length</code>	Maximum length to merge, less equal than the total number of elements available.

**Returns**

End iterator of output sequence.

Definition at line 574 of file `multiway_merge.h`.

References `_GLIBCXX_CALL`.

```
3.6.4.71 template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType > void
__gnu_parallel::multiway_merge_sampling_splitting ( _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end,
_DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair<
_DifferenceType, _DifferenceType > > * __pieces )
```

Sampling based splitting for parallel multiway-merge routine.

Definition at line 1035 of file `multiway_merge.h`.

References `_GLIBCXX_PARALLEL_LENGTH`, `__gnu_parallel::_Settings::get()`, and `__gnu_parallel::_Settings::merge_oversampling`.

```
3.6.4.72 template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >
_RAIterOut __gnu_parallel::multiway_merge_sentinels ( _RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end,
_RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag )
```

Multiway Merge Frontend.

Merge the sequences specified by `seqs_begin` and `__seqs_end` into `__target`. `__seqs_begin` and `__seqs_end` must point to a sequence of pairs. These pairs must contain an iterator to the beginning of a sequence in their first entry and an iterator the `_M_end` of the same sequence in their second entry.

Ties are broken arbitrarily. See `stable_multiway_merge` for a variant that breaks ties by sequence number but is slower.

The first entries of the pairs (i.e. the begin iterators) will be moved forward accordingly.

The output sequence has to provide enough space for all elements that are written to it.

This function will merge the input sequences:

- not stable
- parallel, depending on the input size and Settings
- using sampling for splitting
- using sentinels

You have to take care that the element the `_M_end` iterator points to is readable and contains a value that is greater than any other non-sentinel value in all sequences.

Example:

```
int sequences[10][11];
for (int __i = 0; __i < 10; ++__i)
    for (int __j = 0; __j < 11; ++__j)
        sequences[__i][__j] = __j; // __last one is sentinel!

int __out[33];
std::vector<std::pair<int*> > seqs;
for (int __i = 0; __i < 10; ++__i)
    { seqs.push(std::make_pair<int*>(sequences[__i],
                                    sequences[__i] + 10)) }

multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int*>(), 33);
```

#### Precondition

All input sequences must be sorted.

Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.

For each `__i`, `__seqs_begin[__i].second` must be the end marker of the sequence, but also reference the one more `__sentinel` element.

#### Postcondition

`[__target, return __value)` contains merged `__elements` from the input sequences.

`return __value - __target = min(__length, number of elements in all sequences).`

#### See also

`stable_multiway_merge_sentinels`

## Template Parameters

<code>__RAIterPairIterator</code>	iterator over sequence of pairs of iterators
<code>__RAIterOut</code>	iterator over target sequence
<code>__DifferenceTp</code>	difference type for the sequence
<code>__Compare</code>	strict weak ordering type to compare elements in sequences

## Parameters

<code>__seqs_begin</code>	begin of sequence <code>__sequence</code>
<code>__seqs_end</code>	<code>_M_end</code> of sequence <code>__sequence</code>
<code>__target</code>	target sequence to merge to.
<code>__comp</code>	strict weak ordering to use for element comparison.
<code>__length</code>	Maximum length to merge, possibly larger than the number of elements available.

## Returns

`_M_end` iterator of output sequence

Definition at line 1782 of file `multiway_merge.h`.

References `__sequential_multiway_merge()`, and `_GLIBCXX_CALL`.

```
3.6.4.73 template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp,
typename _Splitter, typename _Compare> _RAIter3 __gnu_parallel::parallel_multiway_merge ( _RAIterIterator
__seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _Splitter __splitter, _DifferenceTp __length, _Compare
__comp, _ThreadIndex __num_threads )
```

Parallel multi-way merge routine.

The `_GLIBCXX_PARALLEL_DECISION` is based on the branching factor and runtime settings.

Must not be called if the number of sequences is 1.

## Template Parameters

<code>__Splitter</code>	functor to split input (either <code>__exact</code> or sampling based)
<code>__stable</code>	Stable merging incurs a performance penalty.
<code>__sentinel</code>	Ignored.

## Parameters

<code>__seqs_begin</code>	Begin iterator of iterator pair input sequence.
<code>__seqs_end</code>	End iterator of iterator pair input sequence.
<code>__target</code>	Begin iterator of output sequence.
<code>__comp</code>	Comparator.
<code>__length</code>	Maximum length to merge, possibly larger than the number of elements available.

## Returns

End iterator of output sequence.

Definition at line 1225 of file `multiway_merge.h`.

References `__is_sorted()`, `_GLIBCXX_CALL`, `_GLIBCXX_PARALLEL_LENGTH`, `__gnu_parallel::_Settings::get()`, `std::make_pair()`, and `__gnu_parallel::_Settings::merge_oversampling`.

Referenced by `__parallel_merge_advance()`.

3.6.4.74 `template<bool __stable, bool __exact, typename _RAIter, typename _Compare > void __gnu_parallel::parallel_sort_mwms  
( _RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads )`

PMWMS main call.

## Parameters

<code>__begin</code>	Begin iterator of sequence.
<code>__end</code>	End iterator of sequence.
<code>__comp</code>	Comparator.
<code>__num_threads</code>	Number of threads to use.

Definition at line 395 of file `multiway_mergesort.h`.

References `_GLIBCXX_CALL`, `__gnu_parallel::PMWMSortingData<_RAIter>::M_num_threads`, `__gnu_parallel::PMWMSortingData<_RAIter>::M_offsets`, `__gnu_parallel::PMWMSortingData<_RAIter>::M_pieces`, `__gnu_parallel::PMWMSortingData<_RAIter>::M_samples`, `__gnu_parallel::PMWMSortingData<_RAIter>::M_source`, `__gnu_parallel::PMWMSortingData<_RAIter>::M_starts`, `__gnu_parallel::PMWMSortingData<_RAIter>::M_temporary`, `__gnu_parallel::Settings::get()`, and `__gnu_parallel::Settings::sort_mwms_oversampling`.

```
3.6.4.75 template<bool __stable, bool __exact, typename _RAIter, typename _Compare> void __gnu_parallel::parallel_sort_mwms_pu ( _PMWMSortingData<_RAIter> * __sd, _Compare & __comp )
```

PMWMS code executed by each thread.

## Parameters

<code>__sd</code>	Pointer to algorithm data.
<code>__comp</code>	Comparator.

Definition at line 308 of file `multiway_mergesort.h`.

References `__gnu_parallel::PMWMSortingData<_RAIter>::M_num_threads`, `__gnu_parallel::PMWMSortingData<_RAIter>::M_pieces`, `__gnu_parallel::PMWMSortingData<_RAIter>::M_source`, `__gnu_parallel::PMWMSortingData<_RAIter>::M_starts`, `__gnu_parallel::PMWMSortingData<_RAIter>::M_temporary`, `__gnu_parallel::Settings::get()`, `std::make_pair()`, `__gnu_parallel::Settings::sort_mwms_oversampling`, and `std::uninitialized_copy()`.

## 3.6.5 Variable Documentation

```
3.6.5.1 const int __gnu_parallel::_CASable_bits [static]
```

Number of bits of `_CASable`.

Definition at line 130 of file `types.h`.

Referenced by `__decode2()`, and `__encode2()`.

```
3.6.5.2 const _CASable __gnu_parallel::_CASable_mask [static]
```

`_CASable` with the right half of bits set to 1.

Definition at line 133 of file `types.h`.

Referenced by `__decode2()`.

## 3.7 \_\_gnu\_pbds Namespace Reference

## Classes

- struct [associative\\_tag](#)



- class [basic\\_branch](#)
- struct [basic\\_branch\\_tag](#)
- class [basic\\_hash\\_table](#)
- struct [basic\\_hash\\_tag](#)
- struct [basic\\_invalidation\\_guarantee](#)
- struct [binary\\_heap\\_tag](#)
- struct [binomial\\_heap\\_tag](#)
- class [cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger](#)
- class [cc\\_hash\\_table](#)
- struct [cc\\_hash\\_tag](#)
- struct [container\\_error](#)
- struct [container\\_tag](#)
- struct [container\\_traits](#)
- struct [container\\_traits\\_base](#)
- struct [container\\_traits\\_base< binary\\_heap\\_tag >](#)
- struct [container\\_traits\\_base< binomial\\_heap\\_tag >](#)
- struct [container\\_traits\\_base< cc\\_hash\\_tag >](#)
- struct [container\\_traits\\_base< gp\\_hash\\_tag >](#)
- struct [container\\_traits\\_base< list\\_update\\_tag >](#)
- struct [container\\_traits\\_base< ov\\_tree\\_tag >](#)
- struct [container\\_traits\\_base< pairing\\_heap\\_tag >](#)
- struct [container\\_traits\\_base< pat\\_trie\\_tag >](#)
- struct [container\\_traits\\_base< rb\\_tree\\_tag >](#)
- struct [container\\_traits\\_base< rc\\_binomial\\_heap\\_tag >](#)
- struct [container\\_traits\\_base< splay\\_tree\\_tag >](#)
- struct [container\\_traits\\_base< thin\\_heap\\_tag >](#)
- class [direct\\_mask\\_range\\_hashing](#)
- class [direct\\_mod\\_range\\_hashing](#)
- class [gp\\_hash\\_table](#)
- struct [gp\\_hash\\_tag](#)
- class [hash\\_exponential\\_size\\_policy](#)
- class [hash\\_load\\_check\\_resize\\_trigger](#)
- class [hash\\_prime\\_size\\_policy](#)
- class [hash\\_standard\\_resize\\_policy](#)
- struct [insert\\_error](#)
- struct [join\\_error](#)
- class [linear\\_probe\\_fn](#)
- class [list\\_update](#)
- struct [list\\_update\\_tag](#)
- class [lu\\_counter\\_policy](#)
- class [lu\\_move\\_to\\_front\\_policy](#)
- struct [null\\_node\\_update](#)
- struct [null\\_type](#)
- struct [ov\\_tree\\_tag](#)
- struct [pairing\\_heap\\_tag](#)
- struct [pat\\_trie\\_tag](#)
- struct [point\\_invalidation\\_guarantee](#)
- class [priority\\_queue](#)
- struct [priority\\_queue\\_tag](#)
- class [quadratic\\_probe\\_fn](#)
- struct [range\\_invalidation\\_guarantee](#)

- struct `rb_tree_tag`
- struct `rc_binomial_heap_tag`
- struct `resize_error`
- class `sample_probe_fn`
- class `sample_range_hashing`
- class `sample_ranged_hash_fn`
- class `sample_ranged_probe_fn`
- class `sample_resize_policy`
- class `sample_resize_trigger`
- class `sample_size_policy`
- class `sample_tree_node_update`
- struct `sample_trie_access_traits`
- class `sample_trie_node_update`
- struct `sample_update_policy`
- struct `sequence_tag`
- struct `splay_tree_tag`
- struct `string_tag`
- struct `thin_heap_tag`
- class `tree`
- class `tree_order_statistics_node_update`
- struct `tree_tag`
- class `trie`
- class `trie_order_statistics_node_update`
- class `trie_prefix_search_node_update`
- struct `trie_string_access_traits`
- struct `trie_tag`
- struct `trivial_iterator_tag`

#### Typedefs

- typedef void `trivial_iterator_difference_type`

#### Functions

- void `__throw_container_error ()`
- void `__throw_insert_error ()`
- void `__throw_join_error ()`
- void `__throw_resize_error ()`

##### 3.7.1 Detailed Description

GNU extensions for policy-based data structures for public use.

### 3.8 `__gnu_profile` Namespace Reference

#### Classes

- class `__container_size_info`
- class `__container_size_stack_info`
- class `__hashfunc_info`
- class `__hashfunc_stack_info`
- class `__list2vector_info`
- class `__map2umap_info`
- class `__map2umap_stack_info`
- class `__object_info_base`
- struct `__reentrance_guard`
- class `__stack_hash`
- class `__stack_info_base`
- class `__trace_base`
- class `__trace_container_size`
- class `__trace_hash_func`
- class `__trace_hashtable_size`
- class `__trace_map2umap`
- class `__trace_vector_size`
- class `__trace_vector_to_list`
- class `__vector2list_info`
- class `__vector2list_stack_info`
- struct `__warning_data`

#### Typedefs

- typedef `std::vector< __cost_factor * > __cost_factor_vector`
- typedef `std::unordered_map< std::string, std::string > __env_t`
- typedef `void * __instruction_address_t`
- typedef `const void * __object_t`
- typedef `std::vector< __instruction_address_t > __stack_npt`
- typedef `__stack_npt * __stack_t`
- typedef `std::vector< __warning_data > __warning_vector_t`

#### Enumerations

- enum `__state_type { __ON, __OFF, __INVALID }`

#### Functions

- `std::size_t __env_to_size_t (const char * __env_var, std::size_t __default_value)`
- `template<typename _InputIterator, typename _Function> _Function __for_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `__stack_t __get_stack ()`
- `template<typename _Container> void __insert_top_n (_Container & __output, const typename _Container::value_type & __value, typename _Container::size_type __n)`
- `bool __is_invalid ()`
- `bool __is_off ()`

- `bool __is_on ()`
- `int __log2 (std::size_t __size)`
- `int __log_magnitude (float __f)`
- `float __map_erase_cost (std::size_t __size)`
- `float __map_find_cost (std::size_t __size)`
- `float __map_insert_cost (std::size_t __size)`
- `std::size_t __max_mem ()`
- `FILE * __open_output_file (const char * __extension)`
- `bool __profcxx_init ()`
- `void __profcxx_init_unconditional ()`
- `void __read_cost_factors ()`
- `template<typename _ForwardIterator, typename _Tp> _ForwardIterator __remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __value)`
- `void __report (void)`
- `void __set_cost_factors ()`
- `void __set_max_mem ()`
- `void __set_max_stack_trace_depth ()`
- `void __set_max_warn_count ()`
- `void __set_trace_path ()`
- `std::size_t __size (__stack_t __stack)`
- `std::size_t __stack_max_depth ()`
- `template<typename _Container> void __top_n (const _Container & __input, _Container & __output, typename _Container::size_type __n)`
- `void __trace_hash_func_construct (const void *)`
- `void __trace_hash_func_destruct (const void *, std::size_t, std::size_t, std::size_t)`
- `void __trace_hash_func_init ()`
- `void __trace_hash_func_report (FILE * __f, __warning_vector_t & __warnings)`
- `void __trace_hashtable_size_construct (const void *, std::size_t)`
- `void __trace_hashtable_size_destruct (const void *, std::size_t, std::size_t)`
- `void __trace_hashtable_size_init ()`
- `void __trace_hashtable_size_report (FILE * __f, __warning_vector_t & __warnings)`
- `void __trace_hashtable_size_resize (const void *, std::size_t, std::size_t)`
- `void __trace_list_to_set_construct (const void *)`
- `void __trace_list_to_set_destruct (const void *)`
- `void __trace_list_to_set_find (const void *, std::size_t)`
- `void __trace_list_to_set_insert (const void *, std::size_t, std::size_t)`
- `void __trace_list_to_set_invalid_operator (const void *)`
- `void __trace_list_to_set_iterate (const void *, std::size_t)`
- `void __trace_list_to_slist_construct (const void *)`
- `void __trace_list_to_slist_destruct (const void *)`
- `void __trace_list_to_slist_init ()`
- `void __trace_list_to_slist_operation (const void *)`
- `void __trace_list_to_slist_report (FILE * __f, __warning_vector_t & __warnings)`
- `void __trace_list_to_slist_rewind (const void *)`
- `void __trace_list_to_vector_construct (const void *)`
- `void __trace_list_to_vector_destruct (const void *)`
- `void __trace_list_to_vector_init ()`
- `void __trace_list_to_vector_insert (const void *, std::size_t, std::size_t)`
- `void __trace_list_to_vector_invalid_operator (const void *)`
- `void __trace_list_to_vector_iterate (const void *, std::size_t)`
- `void __trace_list_to_vector_report (FILE * __f, __warning_vector_t & __warnings)`

- void `__trace_list_to_vector_resize` (const void \*, std::size\_t, std::size\_t)
- void `__trace_map_to_unordered_map_construct` (const void \*)
- void `__trace_map_to_unordered_map_destruct` (const void \*)
- void `__trace_map_to_unordered_map_erase` (const void \*, std::size\_t, std::size\_t)
- void `__trace_map_to_unordered_map_find` (const void \*, std::size\_t)
- void `__trace_map_to_unordered_map_init` ()
- void `__trace_map_to_unordered_map_insert` (const void \*, std::size\_t, std::size\_t)
- void `__trace_map_to_unordered_map_invalidate` (const void \*)
- void `__trace_map_to_unordered_map_iterate` (const void \*, std::size\_t)
- void `__trace_map_to_unordered_map_report` (FILE \* \_\_f, \_\_warning\_vector\_t & \_\_warnings)
- void `__trace_vector_size_construct` (const void \*, std::size\_t)
- void `__trace_vector_size_destruct` (const void \*, std::size\_t, std::size\_t)
- void `__trace_vector_size_init` ()
- void `__trace_vector_size_report` (FILE \*, \_\_warning\_vector\_t &)
- void `__trace_vector_size_resize` (const void \*, std::size\_t, std::size\_t)
- void `__trace_vector_to_list_construct` (const void \*)
- void `__trace_vector_to_list_destruct` (const void \*)
- void `__trace_vector_to_list_find` (const void \*, std::size\_t)
- void `__trace_vector_to_list_init` ()
- void `__trace_vector_to_list_insert` (const void \*, std::size\_t, std::size\_t)
- void `__trace_vector_to_list_invalid_operator` (const void \*)
- void `__trace_vector_to_list_iterate` (const void \*, std::size\_t)
- void `__trace_vector_to_list_report` (FILE \*, \_\_warning\_vector\_t &)
- void `__trace_vector_to_list_resize` (const void \*, std::size\_t, std::size\_t)
- bool `__turn` (\_\_state\_type \_\_s)
- bool `__turn_off` ()
- bool `__turn_on` ()
- void `__write` (FILE \* \_\_f, \_\_stack\_t \_\_stack)
- void `__write_cost_factors` ()
- `_GLIBCXX_PROFILE_DEFINE_DATA` (\_\_state\_type, \_\_state, \_\_INVALID)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (\_\_trace\_hash\_func \*, \_\_S\_hash\_func, 0)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (\_\_trace\_hashtable\_size \*, \_\_S\_hashtable\_size, 0)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (\_\_trace\_map2umap \*, \_\_S\_map2umap, 0)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (\_\_trace\_vector\_size \*, \_\_S\_vector\_size, 0)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (\_\_trace\_vector\_to\_list \*, \_\_S\_vector\_to\_list, 0)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (\_\_trace\_list\_to\_slist \*, \_\_S\_list\_to\_slist, 0)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (\_\_trace\_list\_to\_vector \*, \_\_S\_list\_to\_vector, 0)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (\_\_cost\_factor, \_\_vector\_shift\_cost\_factor, {"\_\_vector\_shift\_cost\_factor", 1.0})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (\_\_cost\_factor, \_\_vector\_iterate\_cost\_factor, {"\_\_vector\_iterate\_cost\_factor", 1.0})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (\_\_cost\_factor, \_\_vector\_resize\_cost\_factor, {"\_\_vector\_resize\_cost\_factor", 1.0})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (\_\_cost\_factor, \_\_list\_shift\_cost\_factor, {"\_\_list\_shift\_cost\_factor", 0.0})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (\_\_cost\_factor, \_\_list\_iterate\_cost\_factor, {"\_\_list\_iterate\_cost\_factor", 10.0})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (\_\_cost\_factor, \_\_list\_resize\_cost\_factor, {"\_\_list\_resize\_cost\_factor", 0.0})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (\_\_cost\_factor, \_\_map\_insert\_cost\_factor, {"\_\_map\_insert\_cost\_factor", 1.5})

- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__map_erase_cost_factor`, {"\_\_map\_erase\_cost\_factor", 1.5})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__map_find_cost_factor`, {"\_\_map\_find\_cost\_factor", 1})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__map_iterate_cost_factor`, {"\_\_map\_iterate\_cost\_factor", 2.3})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__umap_insert_cost_factor`, {"\_\_umap\_insert\_cost\_factor", 12.0})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__umap_erase_cost_factor`, {"\_\_umap\_erase\_cost\_factor", 12.0})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__umap_find_cost_factor`, {"\_\_umap\_find\_cost\_factor", 10.0})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__umap_iterate_cost_factor`, {"\_\_umap\_iterate\_cost\_factor", 1.7})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor_vector` \*, `__cost_factors`, 0)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`const char` \*, `_S_trace_file_name`, `_GLIBCXX_PROFILE_TRACE_PATH_ROOT`)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`std::size_t`, `_S_max_warn_count`, `_GLIBCXX_PROFILE_MAX_WARN_COUNT`)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`std::size_t`, `_S_max_stack_depth`, `_GLIBCXX_PROFILE_MAX_STACK_DEPTH`)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`std::size_t`, `_S_max_mem`, `_GLIBCXX_PROFILE_MEM_PER_DIAGNOSTIC`)
- `_GLIBCXX_PROFILE_DEFINE_UNINIT_DATA` (`__env_t`, `__env`)
- `_GLIBCXX_PROFILE_DEFINE_UNINIT_DATA` (`__gnu_cxx::__mutex`, `__global_lock`)

### 3.8.1 Detailed Description

GNU profile code for public use.

### 3.8.2 Typedef Documentation

#### 3.8.2.1 `typedef std::unordered_map<std::string, std::string> __gnu_profile::__env_t`

Internal environment. Values can be set one of two ways: 1. In config file "var = value". The default config file path is `libstdc++-profile.conf`. 2. By setting process environment variables. For instance, in a Bash shell you can set the unit cost of iterating through a map like this: `export __map_iterate_cost_factor=5.0`. If a value is set both in the input file and through an environment variable, the environment value takes precedence.

Definition at line 65 of file `profiler_trace.h`.

### 3.8.3 Function Documentation

#### 3.8.3.1 `bool __gnu_profile::__profcxx_init( ) [inline]`

This function must be called by each instrumentation point.

The common path is inlined fully.

Definition at line 649 of file `profiler_trace.h`.

### 3.8.3.2 void \_\_gnu\_profile::\_\_report ( void ) [inline]

Final report method, registered with **atexit**.

This can also be called directly by user code, including signal handlers. It is protected against deadlocks by the reentrance guard in profiler.h. However, when called from a signal handler that triggers while within \_\_gnu\_profile (under the guarded zone), no output will be produced.

Definition at line 440 of file profiler\_trace.h.

References std::min().

### 3.8.3.3 \_\_gnu\_profile::\_GLIBCXX\_PROFILE\_DEFINE\_UNINIT\_DATA ( \_\_gnu\_cxx::\_\_mutex , \_\_global\_lock )

Master lock.

## 3.9 \_\_gnu\_sequential Namespace Reference

### 3.9.1 Detailed Description

GNU sequential classes for public use.

## 3.10 abi Namespace Reference

### 3.10.1 Detailed Description

The cross-vendor C++ Application Binary Interface. A namespace alias to \_\_cxxabiv1, but user programs should use the alias 'abi'.

A brief overview of an ABI is given in the libstdc++ FAQ, question 5.8 (you may have a copy of the FAQ locally, or you can view the online version at [http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#5\\_8](http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#5_8)).

GCC subscribes to a cross-vendor ABI for C++, sometimes called the IA64 ABI because it happens to be the native ABI for that platform. It is summarized at <http://www.codesourcery.com/cxx-abi/> along with the current specification.

For users of GCC greater than or equal to 3.x, entry points are available in <cxxabi.h>, which notes, *'It is not normally necessary for user programs to include this header, or use the entry points directly. However, this header is available should that be needed.'*

## 3.11 std Namespace Reference

Namespaces

- [\\_\\_debug](#)
- [\\_\\_detail](#)
- [\\_\\_parallel](#)
- [\\_\\_profile](#)
- [regex\\_constants](#)
- [rel\\_ops](#)
- [tr1](#)
- [tr2](#)

## Classes

- struct [\\_\\_atomic\\_base](#)
- struct [\\_\\_atomic\\_base< \\_PTp \\* >](#)
- struct [\\_\\_atomic\\_flag\\_base](#)
- class [\\_\\_codecvt\\_abstract\\_base](#)
- class [\\_\\_ctype\\_abstract\\_base](#)
- class [\\_\\_has\\_iterator\\_category\\_helper](#)
- class [\\_Deque\\_base](#)
- struct [\\_Deque\\_iterator](#)
- struct [\\_Fwd\\_list\\_base](#)
- struct [\\_Fwd\\_list\\_const\\_iterator](#)
- struct [\\_Fwd\\_list\\_iterator](#)
- struct [\\_Fwd\\_list\\_node](#)
- struct [\\_Fwd\\_list\\_node\\_base](#)
- class [\\_Hashtable](#)
- class [\\_List\\_base](#)
- struct [\\_List\\_const\\_iterator](#)
- struct [\\_List\\_iterator](#)
- struct [\\_List\\_node](#)
- class [\\_Temporary\\_buffer](#)
- struct [\\_Vector\\_base](#)
- class [allocator](#)
- class [allocator< void >](#)
- struct [allocator\\_arg\\_t](#)
- struct [allocator\\_traits](#)
- struct [atomic\\_flag](#)
- class [auto\\_ptr](#)
- struct [auto\\_ptr\\_ref](#)
- class [back\\_insert\\_iterator](#)
- class [bad\\_weak\\_ptr](#)
- class [basic\\_ios](#)
- class [basic\\_regex](#)
- class [basic\\_string](#)
- class [bernoulli\\_distribution](#)
- struct [bidirectional\\_iterator\\_tag](#)
- struct [binary\\_function](#)
- class [binary\\_negate](#)
- class [binder1st](#)
- class [binder2nd](#)
- class [binomial\\_distribution](#)
- class [cauchy\\_distribution](#)
- struct [char\\_traits](#)
- struct [char\\_traits< \\_\\_gnu\\_cxx::character< V, I, S > >](#)
- struct [char\\_traits< char >](#)
- struct [char\\_traits< wchar\\_t >](#)
- class [chi\\_squared\\_distribution](#)
- class [codecvt](#)
- class [codecvt< \\_InternT, \\_ExternT, encoding\\_state >](#)
- class [codecvt< char, char, mbstate\\_t >](#)
- class [codecvt< wchar\\_t, char, mbstate\\_t >](#)



- class `codecvt_base`
- class `codecvt_byname`
- class `collate`
- class `collate_byname`
- class `const_mem_fun1_ref_t`
- class `const_mem_fun1_t`
- class `const_mem_fun_ref_t`
- class `const_mem_fun_t`
- class `ctype`
- class `ctype< char >`
- class `ctype< wchar_t >`
- struct `ctype_base`
- class `ctype_byname`
- class `ctype_byname< char >`
- struct `default_delete`
- struct `default_delete< _Tp[]>`
- class `deque`
- class `discard_block_engine`
- class `discrete_distribution`
- struct `divides`
- class `enable_shared_from_this`
- struct `equal_to`
- class `exponential_distribution`
- class `extreme_value_distribution`
- class `fisher_f_distribution`
- struct `forward_iterator_tag`
- class `forward_list`
- class `fpos`
- class `front_insert_iterator`
- class `gamma_distribution`
- class `geometric_distribution`
- struct `greater`
- struct `greater_equal`
- class `gslice`
- class `gslice_array`
- struct `hash`
- struct `hash< __gnu_cxx::__u16vstring >`
- struct `hash< __gnu_cxx::__u32vstring >`
- struct `hash< __gnu_cxx::__vstring >`
- struct `hash< __gnu_cxx::__wvstring >`
- struct `hash< __gnu_cxx::throw_value_limit >`
- struct `hash< __gnu_cxx::throw_value_random >`
- struct `hash< __shared_ptr< _Tp, _Lp > >`
- struct `hash< _Tp * >`
- struct `hash< bool >`
- struct `hash< char >`
- struct `hash< char16_t >`
- struct `hash< char32_t >`
- struct `hash< double >`
- struct `hash< float >`
- struct `hash< int >`

- struct `hash< long >`
- struct `hash< long double >`
- struct `hash< long long >`
- struct `hash< shared_ptr< _Tp > >`
- struct `hash< short >`
- struct `hash< signed char >`
- struct `hash< string >`
- struct `hash< u16string >`
- struct `hash< u32string >`
- struct `hash< unique_ptr< _Tp, _Dp > >`
- struct `hash< unsigned char >`
- struct `hash< unsigned int >`
- struct `hash< unsigned long >`
- struct `hash< unsigned long long >`
- struct `hash< unsigned short >`
- struct `hash< wchar_t >`
- struct `hash< wstring >`
- struct `hash<::vector< bool, _Alloc > >`
- class `independent_bits_engine`
- class `indirect_array`
- struct `input_iterator_tag`
- class `insert_iterator`
- class `ios_base`
- class `istream_iterator`
- class `istreambuf_iterator`
- struct `iterator`
- struct `iterator_traits< _Tp * >`
- struct `iterator_traits< const _Tp * >`
- struct `less`
- struct `less_equal`
- class `linear_congruential_engine`
- class `list`
- class `locale`
- struct `logical_and`
- struct `logical_not`
- struct `logical_or`
- class `lognormal_distribution`
- class `map`
- class `mask_array`
- class `match_results`
- class `mem_fun1_ref_t`
- class `mem_fun1_t`
- class `mem_fun_ref_t`
- class `mem_fun_t`
- class `mersenne_twister_engine`
- class `messages`
- struct `messages_base`
- class `messages_byname`
- struct `minus`
- struct `modulus`
- class `money_base`

- class [money\\_get](#)
- class [money\\_put](#)
- class [moneypunct](#)
- class [moneypunct\\_byname](#)
- class [move\\_iterator](#)
- class [multimap](#)
- struct [multiplies](#)
- class [multiset](#)
- struct [negate](#)
- class [negative\\_binomial\\_distribution](#)
- class [nested\\_exception](#)
- class [normal\\_distribution](#)
- struct [not\\_equal\\_to](#)
- class [num\\_get](#)
- class [num\\_put](#)
- class [numpunct](#)
- class [numpunct\\_byname](#)
- class [ostream\\_iterator](#)
- class [ostreambuf\\_iterator](#)
- struct [output\\_iterator\\_tag](#)
- struct [owner\\_less](#)
- struct [owner\\_less< shared\\_ptr< \\_Tp > >](#)
- struct [owner\\_less< weak\\_ptr< \\_Tp > >](#)
- struct [pair](#)
- class [piecewise\\_constant\\_distribution](#)
- struct [piecewise\\_construct\\_t](#)
- class [piecewise\\_linear\\_distribution](#)
- struct [plus](#)
- class [pointer\\_to\\_binary\\_function](#)
- class [pointer\\_to\\_unary\\_function](#)
- struct [pointer\\_traits](#)
- struct [pointer\\_traits< \\_Tp \\* >](#)
- class [poisson\\_distribution](#)
- class [priority\\_queue](#)
- class [queue](#)
- struct [random\\_access\\_iterator\\_tag](#)
- class [random\\_device](#)
- class [raw\\_storage\\_iterator](#)
- class [regex\\_error](#)
- class [regex\\_iterator](#)
- class [regex\\_token\\_iterator](#)
- struct [regex\\_traits](#)
- class [reverse\\_iterator](#)
- class [seed\\_seq](#)
- class [set](#)
- class [shared\\_ptr](#)
- class [shuffle\\_order\\_engine](#)
- class [slice](#)
- class [slice\\_array](#)
- class [stack](#)
- class [student\\_t\\_distribution](#)

- class [sub\\_match](#)
- class [time\\_base](#)
- class [time\\_get](#)
- class [time\\_get\\_byname](#)
- class [time\\_put](#)
- class [time\\_put\\_byname](#)
- struct [unary\\_function](#)
- class [unary\\_negate](#)
- class [uniform\\_int\\_distribution](#)
- class [uniform\\_real\\_distribution](#)
- class [unique\\_ptr](#)
- class [unique\\_ptr< \\_Tp\[\], \\_Dp >](#)
- class [unordered\\_map](#)
- class [unordered\\_multimap](#)
- class [unordered\\_multiset](#)
- class [unordered\\_set](#)
- struct [uses\\_allocator](#)
- class [vector](#)
- class [vector< bool, \\_Alloc >](#)
- class [weak\\_ptr](#)
- class [weibull\\_distribution](#)

## Typedefs

- `template<typename _Tp> using \_\_allocator\_base = \_\_gnu\_cxx::new\_allocator< _Tp >`
- `typedef unsigned char \_\_atomic\_flag\_data\_type`
- `typedef FILE \_\_c\_file`
- `typedef __locale_t \_\_c\_locale`
- `typedef __gthread_mutex_t \_\_c\_lock`
- `template<typename _Tp, typename _Hash> using \_\_cache\_default = __not_< __and_< __is_fast_hash< _Hash >, is_default_constructible< _Hash >, is_copy_assignable< _Hash >, __detail::__is_noexcept_hash< _Tp, _Hash >>>>`
- `template<typename _Alloc> using \_\_check\_copy\_constructible = __allow_copy_cons< __is_copy_insertable< ↵ _Alloc >::value >`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> using \_\_sub\_match\_string = basic\_string< type-name iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc >`
- `template<typename _Key, typename _Tp, typename _Hash = hash< _Key >, typename _Pred = std::equal_to< _Key >, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __umap_traits<__cache_default< _Key, _Hash >::value>> using \_\_umap\_hashtable = \_Hashtable< _Key, std::pair< const _Key, _Tp >, _Alloc, __detail::__Select1st, _Pred, _Hash, __detail::__Mod_range_hashing, __detail::__Default_ranged_hash, __detail::__Prime_rehash_policy, _Tr >`
- `template<bool _Cache> using \_\_umap\_traits = \_\_detail::\_\_Hashtable\_traits< _Cache, false, true >`
- `template<typename _Key, typename _Tp, typename _Hash = hash< _Key >, typename _Pred = std::equal_to< _Key >, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __ummap_traits<__cache_default< _Key, _Hash >::value>> using \_\_ummap\_hashtable = \_Hashtable< _Key, std::pair< const _Key, _Tp >, _Alloc, __detail::__Select1st, _Pred, _Hash, __detail::__Mod_range_hashing, __detail::__Default_ranged_hash, __detail::__Prime_rehash_policy, _Tr >`
- `template<bool _Cache> using \_\_ummap\_traits = \_\_detail::\_\_Hashtable\_traits< _Cache, false, false >`
- `template<typename _Value, typename _Hash = hash< _Value >, typename _Pred = std::equal_to< _Value >, typename _Alloc = std::↵ allocator< _Value >, typename _Tr = __umset_traits<__cache_default< _Value, _Hash >::value>> using \_\_umset\_hashtable = \_Hashtable< _Value, _Value, _Alloc, __detail::__Identity, _Pred, _Hash, __detail::__Mod_range_hashing, ↵ __detail::__Default_ranged_hash, __detail::__Prime_rehash_policy, _Tr >`
- `template<bool _Cache> using \_\_umset\_traits = \_\_detail::\_\_Hashtable\_traits< _Cache, true, false >`

- `template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __uset_traits<__cache_default<_Value, _Hash>::value>> using __uset_hashtable = __Hashtable<_Value, _Value, _Alloc, __detail::_Identity, _Pred, _Hash, __detail::_Mod_range_hashing, __detail::_Default_ranged_hash, __detail::_Prime_rehash_policy, _Tr>`
- `template<bool _Cache> using __uset_traits = __detail::_Hashtable_traits<_Cache, true, true>`
- `typedef unsigned long _Bit_type`
- `template<typename _InIter > using _RequireInputIter = typename enable_if< is_convertible< typename iterator_traits<_InIter>::iterator_category, input_iterator_tag>::value>::type`
- `typedef __atomic_base< char > atomic_char`
- `typedef __atomic_base< char16_t > atomic_char16_t`
- `typedef __atomic_base< char32_t > atomic_char32_t`
- `typedef __atomic_base< int > atomic_int`
- `typedef __atomic_base< int_fast16_t > atomic_int_fast16_t`
- `typedef __atomic_base< int_fast32_t > atomic_int_fast32_t`
- `typedef __atomic_base< int_fast64_t > atomic_int_fast64_t`
- `typedef __atomic_base< int_fast8_t > atomic_int_fast8_t`
- `typedef __atomic_base< int_least16_t > atomic_int_least16_t`
- `typedef __atomic_base< int_least32_t > atomic_int_least32_t`
- `typedef __atomic_base< int_least64_t > atomic_int_least64_t`
- `typedef __atomic_base< int_least8_t > atomic_int_least8_t`
- `typedef __atomic_base< intmax_t > atomic_intmax_t`
- `typedef __atomic_base< intptr_t > atomic_intptr_t`
- `typedef __atomic_base< long long > atomic_llong`
- `typedef __atomic_base< long > atomic_long`
- `typedef __atomic_base< ptrdiff_t > atomic_ptrdiff_t`
- `typedef __atomic_base< signed char > atomic_schar`
- `typedef __atomic_base< short > atomic_short`
- `typedef __atomic_base< size_t > atomic_size_t`
- `typedef __atomic_base< unsigned char > atomic_uchar`
- `typedef __atomic_base< unsigned int > atomic_uint`
- `typedef __atomic_base< uint_fast16_t > atomic_uint_fast16_t`
- `typedef __atomic_base< uint_fast32_t > atomic_uint_fast32_t`
- `typedef __atomic_base< uint_fast64_t > atomic_uint_fast64_t`
- `typedef __atomic_base< uint_fast8_t > atomic_uint_fast8_t`
- `typedef __atomic_base< uint_least16_t > atomic_uint_least16_t`
- `typedef __atomic_base< uint_least32_t > atomic_uint_least32_t`
- `typedef __atomic_base< uint_least64_t > atomic_uint_least64_t`
- `typedef __atomic_base< uint_least8_t > atomic_uint_least8_t`
- `typedef __atomic_base< uintmax_t > atomic_uintmax_t`
- `typedef __atomic_base< uintptr_t > atomic_uintptr_t`
- `typedef __atomic_base< unsigned long long > atomic_ullong`
- `typedef __atomic_base< unsigned long > atomic_ulong`
- `typedef __atomic_base< unsigned short > atomic_ushort`
- `typedef __atomic_base< wchar_t > atomic_wchar_t`
- `typedef match_results< const char * > cmatch`
- `typedef regex_iterator< const char * > cregex_iterator`
- `typedef regex_token_iterator< const char * > cregex_token_iterator`
- `typedef sub_match< const char * > csub_match`
- `typedef minstd_rand0 default_random_engine`
- `typedef shuffle_order_engine< minstd_rand0, 256 > knuth_b`
- `typedef enum std::memory_order memory_order`

- typedef [linear\\_congruential\\_engine](#)< uint\_fast32\_t, 48271UL, 0UL, 2147483647UL > [minstd\\_rand](#)
- typedef [linear\\_congruential\\_engine](#)< uint\_fast32\_t, 16807UL, 0UL, 2147483647UL > [minstd\\_rand0](#)
- typedef [mersenne\\_twister\\_engine](#)< uint\_fast32\_t, 32, 624, 397, 31, 0x9908b0dFUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > [mt19937](#)
- typedef [mersenne\\_twister\\_engine](#)< uint\_fast64\_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xffff7eee00000000ULL, 43, 6364136223846793005ULL > [mt19937\\_64](#)
- typedef \_\_PTRDIFF\_TYPE\_\_ **ptrdiff\_t**
- typedef [discard\\_block\\_engine](#)< ranlux24\_base, 223, 23 > **ranlux24**
- typedef subtract\_with\_carry\_engine< uint\_fast32\_t, 24, 10, 24 > **ranlux24\_base**
- typedef [discard\\_block\\_engine](#)< ranlux48\_base, 389, 11 > **ranlux48**
- typedef subtract\_with\_carry\_engine< uint\_fast64\_t, 48, 5, 12 > **ranlux48\_base**
- typedef [basic\\_regex](#)< char > **regex**
- typedef \_\_SIZE\_TYPE\_\_ **size\_t**
- typedef [match\\_results](#)< string::const\_iterator > **smatch**
- typedef [regex\\_iterator](#)< string::const\_iterator > **sregex\_iterator**
- typedef [regex\\_token\\_iterator](#)< string::const\_iterator > **sregex\_token\_iterator**
- typedef [sub\\_match](#)< string::const\_iterator > **ssub\_match**
- typedef long long **streamoff**
- typedef [fpos](#)< mbstate\_t > **streampos**
- typedef ptrdiff\_t **streamsize**
- typedef [basic\\_string](#)< char > **string**
- typedef [fpos](#)< mbstate\_t > **u16streampos**
- typedef [basic\\_string](#)< char16\_t > **u16string**
- typedef [fpos](#)< mbstate\_t > **u32streampos**
- typedef [basic\\_string](#)< char32\_t > **u32string**
- typedef [match\\_results](#)< const wchar\_t \* > **wcmatch**
- typedef [regex\\_iterator](#)< const wchar\_t \* > **wcregex\_iterator**
- typedef [regex\\_token\\_iterator](#)< const wchar\_t \* > **wcregex\_token\_iterator**
- typedef [sub\\_match](#)< const wchar\_t \* > **wcsub\_match**
- typedef [basic\\_regex](#)< wchar\_t > **wregex**
- typedef [match\\_results](#)< wstring::const\_iterator > **wsmatch**
- typedef [regex\\_iterator](#)< wstring::const\_iterator > **wsregex\_iterator**
- typedef [regex\\_token\\_iterator](#)< wstring::const\_iterator > **wsregex\_token\_iterator**
- typedef [sub\\_match](#)< wstring::const\_iterator > **wssub\_match**
- typedef [fpos](#)< mbstate\_t > **wstreampos**
- typedef [basic\\_string](#)< wchar\_t > **wstring**

#### Enumerations

- enum { **\_S\_threshold** }
- enum { **\_S\_chunk\_size** }
- enum { **\_S\_word\_bit** }
- enum **\_\_memory\_order\_modifier** { **\_\_memory\_order\_mask**, **\_\_memory\_order\_modifier\_mask**, **\_\_memory\_order\_hle\_acquire**, **\_\_memory\_order\_hle\_release** }
- enum **\_ios\_Fmtflags** { **\_S\_boolalpha**, **\_S\_dec**, **\_S\_fixed**, **\_S\_hex**, **\_S\_internal**, **\_S\_left**, **\_S\_oct**, **\_S\_right**, **\_S\_scientific**, **\_S\_showbase**, **\_S\_showpoint**, **\_S\_showpos**, **\_S\_skipws**, **\_S\_unitbuf**, **\_S\_uppercase**, **\_S\_adjustfield**, **\_S\_basefield**, **\_S\_floatfield**, **\_S\_ios\_fmtflags\_end** }
- enum **\_ios\_iostate** { **\_S\_goodbit**, **\_S\_badbit**, **\_S\_eofbit**, **\_S\_failbit**, **\_S\_ios\_iostate\_end** }
- enum **\_ios\_Openmode** { **\_S\_app**, **\_S\_ate**, **\_S\_bin**, **\_S\_in**, **\_S\_out**, **\_S\_trunc**, **\_S\_ios\_openmode\_end** }
- enum **\_ios\_Seekdir** { **\_S\_beg**, **\_S\_cur**, **\_S\_end**, **\_S\_ios\_seekdir\_end** }

- enum `_Rb_tree_color` { `_S_red`, `_S_black` }
- enum `errc` { `address_family_not_supported`, `address_in_use`, `address_not_available`, `already_connected`, `argument_list_too_long`, `argument_out_of_domain`, `bad_address`, `bad_file_descriptor`, `broken_↵`  
`pipe`, `connection_aborted`, `connection_already_in_progress`, `connection_refused`, `connection_reset`,  
`cross_device_link`, `destination_address_required`, `device_or_resource_busy`, `directory_not_empty`,  
`executable_format_error`, `file_exists`, `file_too_large`, `filename_too_long`, `function_not_supported`, `host_↵`  
`unreachable`, `illegal_byte_sequence`, `inappropriate_io_control_operation`, `interrupted`, `invalid_argument`,  
`invalid_seek`, `io_error`, `is_a_directory`, `message_size`, `network_down`, `network_reset`, `network_↵`  
`unreachable`, `no_buffer_space`, `no_child_process`, `no_lock_available`, `no_message`, `no_protocol_option`,  
`no_space_on_device`, `no_such_device_or_address`, `no_such_device`, `no_such_file_or_directory`, `no_↵`  
`_such_process`, `not_a_directory`, `not_a_socket`, `not_connected`, `not_enough_memory`, `operation_in_↵`  
`progress`, `operation_not_permitted`, `operation_not_supported`, `operation_would_block`, `permission_↵`  
`denied`, `protocol_not_supported`, `read_only_file_system`, `resource_deadlock_would_occur`, `resource_↵`  
`unavailable_try_again`, `result_out_of_range`, `timed_out`, `too_many_files_open_in_system`, `too_many_↵`  
`files_open`, `too_many_links`, `too_many_symbolic_link_levels`, `wrong_protocol_type` }
- enum `memory_order` { `memory_order_relaxed`, `memory_order_consume`, `memory_order_acquire`,  
`memory_order_release`, `memory_order_acq_rel`, `memory_order_seq_cst` }

## Functions

- template<typename `_CharT` > `_CharT * __add_grouping` (`_CharT * __s`, `_CharT __sep`, `const char * __gbeg`, `size_t`  
`__gsize`, `const _CharT * __first`, `const _CharT * __last`)
- template<typename `_Tp` > `_Tp * __addressof` (`_Tp & __r`) noexcept
- template<typename `_RandomAccessIterator` , typename `_Distance` , typename `_Tp` > void `__adjust_heap` (`_RandomAccess_↵`  
`Iterator __first`, `_Distance __holeIndex`, `_Distance __len`, `_Tp __value`)
- template<typename `_RandomAccessIterator` , typename `_Distance` , typename `_Tp` , typename `_Compare` > void `__adjust_heap`  
(`_RandomAccessIterator __first`, `_Distance __holeIndex`, `_Distance __len`, `_Tp __value`, `_Compare __comp`)
- template<typename `_InputIterator` , typename `_Distance` > void `__advance` (`_InputIterator & __i`, `_Distance __n`, `input_↵`  
`iterator_tag`)
- template<typename `_BidirectionalIterator` , typename `_Distance` > void `__advance` (`_BidirectionalIterator & __i`, `_Distance ↵`  
`__n`, `bidirectional_iterator_tag`)
- template<typename `_RandomAccessIterator` , typename `_Distance` > void `__advance` (`_RandomAccessIterator & __i`, `↵`  
`_Distance __n`, `random_access_iterator_tag`)
- template<typename `_Alloc` > void `__alloc_on_copy` (`_Alloc & __one`, `const _Alloc & __two`)
- template<typename `_Alloc` > `_Alloc __alloc_on_copy` (`const _Alloc & __a`)
- template<typename `_Alloc` > void `__alloc_on_move` (`_Alloc & __one`, `_Alloc & __two`)
- template<typename `_Alloc` > void `__alloc_on_swap` (`_Alloc & __one`, `_Alloc & __two`)
- template<typename `_Tp` , `_Lock_policy _Lp`, typename `_Alloc` , typename... `_Args`> `__shared_ptr< _Tp, _Lp > __allocate_↵`  
`shared` (`const _Alloc & __a`, `_Args &&... __args`)
- template<typename `_Facet` > const `_Facet & __check_facet` (`const _Facet * __f`)
- template<typename `_RandomAccessIterator` , typename `_Distance` > void `__chunk_insertion_sort` (`_RandomAccessIterator`  
`__first`, `_RandomAccessIterator __last`, `_Distance __chunk_size`)
- template<typename `_RandomAccessIterator` , typename `_Distance` , typename `_Compare` > void `__chunk_insertion_sort` (`↵`  
`RandomAccessIterator __first`, `_RandomAccessIterator __last`, `_Distance __chunk_size`, `_Compare __comp`)
- constexpr `memory_order __cmpexch_failure_order` (`memory_order __m`) noexcept
- constexpr `memory_order __cmpexch_failure_order2` (`memory_order __m`) noexcept
- int `__convert_from_v` (`const __c_locale & __cloc`, `__attribute__((__unused__))`, `char * __out`, `const int __size ↵`  
`__attribute__((__unused__))`, `const char * __fmt`,...)
- template<typename `_Tp` > void `__convert_to_v` (`const char *`, `_Tp &`, `ios_base::iostate &`, `const __c_locale &`) throw  
( )
- template<> void `__convert_to_v` (`const char *`, `float &`, `ios_base::iostate &`, `const __c_locale &`) throw ( )
- template<> void `__convert_to_v` (`const char *`, `double &`, `ios_base::iostate &`, `const __c_locale &`) throw ( )

- `template<> void __convert_to_v (const char *, long double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<bool _IsMove, typename _II, typename _OI> _OI __copy_move_a (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _CharT> __gnu_cxx::__enable_if<__is_char<_CharT>::__value, ostreambuf_iterator<_CharT> >::__type __copy_move_a2 (_CharT *__first, _CharT *__last, ostreambuf_iterator<_CharT> __result)`
- `template<bool _IsMove, typename _CharT> __gnu_cxx::__enable_if<__is_char<_CharT>::__value, ostreambuf_iterator<_CharT> >::__type __copy_move_a2 (const _CharT *__first, const _CharT *__last, ostreambuf_iterator<_CharT> __result)`
- `template<bool _IsMove, typename _CharT> __gnu_cxx::__enable_if<__is_char<_CharT>::__value, _CharT * >::__type __copy_move_a2 (istreambuf_iterator<_CharT> __first, istreambuf_iterator<_CharT> __last, _CharT *__result)`
- `template<bool _IsMove, typename _CharT> __gnu_cxx::__enable_if<__is_char<_CharT>::__value, ostreambuf_iterator<_CharT, char_traits<_CharT> > >::__type __copy_move_a2 (_CharT *, _CharT *, ostreambuf_iterator<_CharT, char_traits<_CharT> >)`
- `template<bool _IsMove, typename _CharT> __gnu_cxx::__enable_if<__is_char<_CharT>::__value, ostreambuf_iterator<_CharT, char_traits<_CharT> > >::__type __copy_move_a2 (const _CharT *, const _CharT *, ostreambuf_iterator<_CharT, char_traits<_CharT> >)`
- `template<bool _IsMove, typename _CharT> __gnu_cxx::__enable_if<__is_char<_CharT>::__value, _CharT * >::__type __copy_move_a2 (istreambuf_iterator<_CharT, char_traits<_CharT> >, istreambuf_iterator<_CharT, char_traits<_CharT> >, _CharT *)`
- `template<bool _IsMove, typename _II, typename _OI> _OI __copy_move_a2 (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2> _BI2 __copy_move_backward_a (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2> _BI2 __copy_move_backward_a2 (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator> _OutputIterator __copy_n (_InputIterator __first, _Size __n, _OutputIterator __result, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Size, typename _OutputIterator> _OutputIterator __copy_n (_RandomAccessIterator __first, _Size __n, _OutputIterator __result, random_access_iterator_tag)`
- `size_t __deque_buf_size (size_t __size)`
- `template<typename _InputIterator> iterator_traits<_InputIterator>::difference_type __distance (_InputIterator __first, _InputIterator __last, input_iterator_tag)`
- `template<typename _RandomAccessIterator> iterator_traits<_RandomAccessIterator>::difference_type __distance (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _Alloc> void __do_alloc_on_copy (_Alloc &__one, const _Alloc &__two, true_type)`
- `template<typename _Alloc> void __do_alloc_on_copy (_Alloc &, const _Alloc &, false_type)`
- `template<typename _Alloc> void __do_alloc_on_move (_Alloc &__one, _Alloc &__two, true_type)`
- `template<typename _Alloc> void __do_alloc_on_move (_Alloc &, _Alloc &, false_type)`
- `template<typename _Alloc> void __do_alloc_on_swap (_Alloc &__one, _Alloc &__two, true_type)`
- `template<typename _Alloc> void __do_alloc_on_swap (_Alloc &, _Alloc &, false_type)`
- `template<_Lock_policy _Lp, typename _Tp1, typename _Tp2> void __enable_shared_from_this_helper (const __shared_count<_Lp> &, const __enable_shared_from_this<_Tp1, _Lp> *, const _Tp2 *) noexcept`
- `template<typename _Tp1, typename _Tp2> void __enable_shared_from_this_helper (const __shared_count<> &, const enable_shared_from_this<_Tp1> *, const _Tp2 *) noexcept`
- `template<_Lock_policy _Lp> void __enable_shared_from_this_helper (const __shared_count<_Lp> &,...) noexcept`
- `template<typename _II1, typename _II2> bool __equal_aux (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _ForwardIterator, typename _Tp> __gnu_cxx::__enable_if<!__is_scalar<_Tp>::__value, void >::__type __fill_a (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _ForwardIterator, typename _Tp> __gnu_cxx::__enable_if<__is_scalar<_Tp>::__value, void >::__type __fill_a (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`



- `template<typename _Tp > __gnu_cxx::__enable_if< __is_byte< _Tp >::__value, void >::__type __fill_a ( _Tp * __first, _Tp * __last, const _Tp & __c)`
- `void __fill_bvector ( _Bit_iterator __first, _Bit_iterator __last, bool __x)`
- `template<typename _OutputIterator, typename _Size, typename _Tp > __gnu_cxx::__enable_if<! __is_scalar< _Tp >::__value, _OutputIterator >::__type __fill_n_a ( _OutputIterator __first, _Size __n, const _Tp & __value)`
- `template<typename _OutputIterator, typename _Size, typename _Tp > __gnu_cxx::__enable_if< __is_scalar< _Tp >::__value, _OutputIterator >::__type __fill_n_a ( _OutputIterator __first, _Size __n, const _Tp & __value)`
- `template<typename _Size, typename _Tp > __gnu_cxx::__enable_if< __is_byte< _Tp >::__value, _Tp * >::__type __fill_n_a ( _Tp * __first, _Size __n, const _Tp & __c)`
- `template<typename _RandomAccessIterator > void __final_insertion_sort ( _RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare > void __final_insertion_sort ( _RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _Tp > _InputIterator __find ( _InputIterator __first, _InputIterator __last, const _Tp & __val, input\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Tp > _RandomAccessIterator __find ( _RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp & __val, random\_access\_iterator\_tag)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 > _ForwardIterator1 __find_end ( _ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, forward\_iterator\_tag, forward\_iterator\_tag)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate > _ForwardIterator1 __find_end ( _ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, forward\_iterator\_tag, forward\_iterator\_tag, _BinaryPredicate __comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2 > _BidirectionalIterator1 __find_end ( _BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, bidirectional\_iterator\_tag, bidirectional\_iterator\_tag)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BinaryPredicate > _BidirectionalIterator1 __find_end ( _BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, bidirectional\_iterator\_tag, bidirectional\_iterator\_tag, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _Predicate > _InputIterator __find_if ( _InputIterator __first, _InputIterator __last, _Predicate __pred, input\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Predicate > _RandomAccessIterator __find_if ( _RandomAccessIterator __first, _RandomAccessIterator __last, _Predicate __pred, random\_access\_iterator\_tag)`
- `template<typename _InputIterator, typename _Predicate > _InputIterator __find_if_not ( _InputIterator __first, _InputIterator __last, _Predicate __pred, input\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Predicate > _RandomAccessIterator __find_if_not ( _RandomAccessIterator __first, _RandomAccessIterator __last, _Predicate __pred, random\_access\_iterator\_tag)`
- `template<typename _InputIterator, typename _Predicate > _InputIterator __find_if_not ( _InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate, typename _Distance > _InputIterator __find_if_not_n ( _InputIterator __first, _Distance & __len, _Predicate __pred)`
- `template<typename _EuclideanRingElement > _EuclideanRingElement __gcd ( _EuclideanRingElement __m, _EuclideanRingElement __n)`
- `template<typename _Ex > const nested\_exception * __get_nested_exception (const _Ex & __ex)`
- `template<typename _RandomAccessIterator > void __heap_select ( _RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare > void __heap_select ( _RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Tp > size_t __iconv_adaptor (size_t (* __func)(iconv_t, _Tp, size_t *, char **, size_t *), iconv_t __cd, char ** __inbuf, size_t * __inbytes, char ** __outbuf, size_t * __outbytes)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Distance > _ForwardIterator __inplace_stable_partition ( _ForwardIterator __first, _Predicate __pred, _Distance __len)`

- `template<typename _RandomAccessIterator> void \_\_inplace\_stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare> void \_\_inplace\_stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator> void \_\_insertion\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare> void \_\_insertion\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Size> void \_\_introsselect (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Size __depth_limit)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare> void \_\_introsselect (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Size> void \_\_introsort\_loop (_RandomAccessIterator __first, _RandomAccessIterator __last, _Size __depth_limit)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare> void \_\_introsort\_loop (_RandomAccessIterator __first, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance> bool \_\_is\_heap (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator, typename _Compare, typename _Distance> bool \_\_is\_heap (_RandomAccessIterator __first, _Compare __comp, _Distance __n)`
- `template<typename _RandomAccessIterator> bool \_\_is\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare> bool \_\_is\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance> _Distance \_\_is\_heap\_until (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare> _Distance \_\_is\_heap\_until (_RandomAccessIterator __first, _Distance __n, _Compare __comp)`
- `template<typename _Iter> iterator_traits<_Iter>::iterator_category \_\_iterator\_category (const _Iter &)`
- `template<typename _I1, typename _I2> bool \_\_lexicographical\_compare\_aux (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2)`
- `constexpr int \_\_lg (int __n)`
- `constexpr unsigned \_\_lg (unsigned __n)`
- `constexpr long \_\_lg (long __n)`
- `constexpr unsigned long \_\_lg (unsigned long __n)`
- `constexpr long long \_\_lg (long long __n)`
- `constexpr unsigned long long \_\_lg (unsigned long long __n)`
- `template<typename _Iterator, typename _ReturnType = typename conditional<__move_if_noexcept_cond <typename iterator_traits<_Iterator>::value_type>::value, _Iterator, move_iterator<_Iterator>>::type> _ReturnType \_\_make\_move\_if\_noexcept\_iterator (_Iterator __i)`
- `template<typename _Tp, _Lock_policy _Lp, typename... _Args> __shared_ptr<_Tp, _Lp> \_\_make\_shared (_Args &&... __args)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer> void \_\_merge\_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _Compare> void \_\_merge\_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)`
- `template<typename _RandomAccessIterator1, typename _RandomAccessIterator2, typename _Distance> void \_\_merge\_sort\_loop (_RandomAccessIterator1 __first, _RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance __step_size)`

- `template<typename _RandomAccessIterator1, typename _RandomAccessIterator2, typename _Distance, typename _Compare> void`  
`__merge_sort_loop` (`_RandomAccessIterator1 __first`, `_RandomAccessIterator1 __last`, `_RandomAccessIterator2 __result`, `_Distance __step_size`, `_Compare __comp`)
- `template<typename _RandomAccessIterator, typename _Pointer> void` `__merge_sort_with_buffer` (`_RandomAccessIterator __first`, `_RandomAccessIterator __last`, `_Pointer __buffer`)
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Compare> void` `__merge_sort_with_buffer` (`_RandomAccessIterator __first`, `_RandomAccessIterator __last`, `_Pointer __buffer`, `_Compare __comp`)
- `template<typename _BidirectionalIterator, typename _Distance> void` `__merge_without_buffer` (`_BidirectionalIterator __first`, `_BidirectionalIterator __middle`, `_BidirectionalIterator __last`, `_Distance __len1`, `_Distance __len2`)
- `template<typename _BidirectionalIterator, typename _Distance, typename _Compare> void` `__merge_without_buffer` (`_BidirectionalIterator __first`, `_BidirectionalIterator __middle`, `_BidirectionalIterator __last`, `_Distance __len1`, `_Distance __len2`, `_Compare __comp`)
- `template<typename _Iterator> _Miter_base<_Iterator>::iterator_type` `__miter_base` (`_Iterator __it`)
- `template<typename _Iterator> void` `__move_median_to_first` (`_Iterator __result`, `_Iterator __a`, `_Iterator __b`, `_Iterator __c`)
- `template<typename _Iterator, typename _Compare> void` `__move_median_to_first` (`_Iterator __result`, `_Iterator __a`, `_Iterator __b`, `_Iterator __c`, `_Compare __comp`)
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator> _OutputIterator` `__move_merge` (`_InputIterator1 __first1`, `_InputIterator1 __last1`, `_InputIterator2 __first2`, `_InputIterator2 __last2`, `_OutputIterator __result`)
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare> _OutputIterator` `__move_merge` (`_InputIterator1 __first1`, `_InputIterator1 __last1`, `_InputIterator2 __first2`, `_InputIterator2 __last2`, `_OutputIterator __result`, `_Compare __comp`)
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator> void` `__move_merge_adaptive` (`_InputIterator1 __first1`, `_InputIterator1 __last1`, `_InputIterator2 __first2`, `_InputIterator2 __last2`, `_OutputIterator __result`)
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare> void` `__move_merge_adaptive` (`_InputIterator1 __first1`, `_InputIterator1 __last1`, `_InputIterator2 __first2`, `_InputIterator2 __last2`, `_OutputIterator __result`, `_Compare __comp`)
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BidirectionalIterator3> void` `__move_merge_adaptive_backward` (`_BidirectionalIterator1 __first1`, `_BidirectionalIterator1 __last1`, `_BidirectionalIterator2 __first2`, `_BidirectionalIterator2 __last2`, `_BidirectionalIterator3 __result`)
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BidirectionalIterator3, typename _Compare> void` `__move_merge_adaptive_backward` (`_BidirectionalIterator1 __first1`, `_BidirectionalIterator1 __last1`, `_BidirectionalIterator2 __first2`, `_BidirectionalIterator2 __last2`, `_BidirectionalIterator3 __result`, `_Compare __comp`)
- `template<typename _Iterator> _Niter_base<_Iterator>::iterator_type` `__niter_base` (`_Iterator __it`)
- `template<typename _CharT, typename _Traits> void` `__ostream_fill` (`basic_ostream<_CharT, _Traits> &__out`, `streamsize __n`)
- `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> &` `__ostream_insert` (`basic_ostream<_CharT, _Traits> &__out`, `const _CharT *__s`, `streamsize __n`)
- `template<typename _CharT, typename _Traits> void` `__ostream_write` (`basic_ostream<_CharT, _Traits> &__out`, `const _CharT *__s`, `streamsize __n`)
- `template<typename _ForwardIterator, typename _Predicate> _ForwardIterator` `__partition` (`_ForwardIterator __first`, `_ForwardIterator __last`, `_Predicate __pred`, `forward_iterator_tag`)
- `template<typename _BidirectionalIterator, typename _Predicate> _BidirectionalIterator` `__partition` (`_BidirectionalIterator __first`, `_BidirectionalIterator __last`, `_Predicate __pred`, `bidirectional_iterator_tag`)
- `template<typename _RandomAccessIterator> void` `__pop_heap` (`_RandomAccessIterator __first`, `_RandomAccessIterator __last`, `_RandomAccessIterator __result`)
- `template<typename _RandomAccessIterator, typename _Compare> void` `__pop_heap` (`_RandomAccessIterator __first`, `_RandomAccessIterator __last`, `_RandomAccessIterator __result`, `_Compare __comp`)
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp> void` `__push_heap` (`_RandomAccessIterator __first`, `_Distance __holeIndex`, `_Distance __topIndex`, `_Tp __value`)

- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare> void __push_heap (↵  
_RandomAccessIterator __first, _Distance __holeIndex, _Distance __topIndex, _Tp __value, _Compare __comp)`
- `template<typename _BidirectionalIterator> void __reverse (_BidirectionalIterator __first, _BidirectionalIterator __last,  
bidirectional\_iterator\_tag)`
- `template<typename _RandomAccessIterator> void __reverse (_RandomAccessIterator __first, _RandomAccessIterator  
__last, random\_access\_iterator\_tag)`
- `template<typename _ForwardIterator> void __rotate (_ForwardIterator __first, _ForwardIterator __middle, _Forward↵  
Iterator __last, forward\_iterator\_tag)`
- `template<typename _BidirectionalIterator> void __rotate (_BidirectionalIterator __first, _BidirectionalIterator __middle,  
_BidirectionalIterator __last, bidirectional\_iterator\_tag)`
- `template<typename _RandomAccessIterator> void __rotate (_RandomAccessIterator __first, _RandomAccessIterator  
__middle, _RandomAccessIterator __last, random\_access\_iterator\_tag)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _Distance> _BidirectionalIterator1 ↵  
rotate\_adaptive (_BidirectionalIterator1 __first, _BidirectionalIterator1 __middle, _BidirectionalIterator1 __last, ↵  
_Distance __len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance __buffer_size)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp> _ForwardIterator __search_n (_ForwardIterator  
__first, _ForwardIterator __last, _Integer __count, const _Tp &__val, std::forward\_iterator\_tag)`
- `template<typename _RandomAccessIter, typename _Integer, typename _Tp> _RandomAccessIter __search_n (_Random↵  
AccessIter __first, _RandomAccessIter __last, _Integer __count, const _Tp &__val, std::random\_access\_↵  
iterator\_tag)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate> _ForwardIterator ↵  
\_\_search\_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _Binary↵  
Predicate __binary_pred, std::forward\_iterator\_tag)`
- `template<typename _RandomAccessIter, typename _Integer, typename _Tp, typename _BinaryPredicate> _RandomAccessIter  
__search_n (_RandomAccessIter __first, _RandomAccessIter __last, _Integer __count, const _Tp &__val, ↵  
_BinaryPredicate __binary_pred, std::random\_access\_iterator\_tag)`
- `template<typename _ForwardIterator, typename _Pointer, typename _Predicate, typename _Distance> _ForwardIterator ↵  
stable\_partition\_adaptive (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, _Distance __len,  
_Pointer __buffer, _Distance __buffer_size)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Distance> void __stable_sort_adaptive (↵  
_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Distance, typename _Compare> void __stable↵  
sort\_adaptive (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Distance ↵  
__buffer_size, _Compare __comp)`
- `void __throw_bad_alloc (void) __attribute__((__noreturn__))`
- `void __throw_bad_cast (void) __attribute__((__noreturn__))`
- `void __throw_bad_exception (void) __attribute__((__noreturn__))`
- `void __throw_bad_function_call () __attribute__((__noreturn__))`
- `void __throw_bad_typeid (void) __attribute__((__noreturn__))`
- `void __throw_bad_weak_ptr ()`
- `void __throw_domain_error (const char *) __attribute__((__noreturn__))`
- `void __throw_future_error (int) __attribute__((__noreturn__))`
- `void __throw_invalid_argument (const char *) __attribute__((__noreturn__))`
- `void __throw_ios_failure (const char *) __attribute__((__noreturn__))`
- `void __throw_length_error (const char *) __attribute__((__noreturn__))`
- `void __throw_logic_error (const char *) __attribute__((__noreturn__))`
- `void __throw_out_of_range (const char *) __attribute__((__noreturn__))`
- `void __throw_overflow_error (const char *) __attribute__((__noreturn__))`
- `void __throw_range_error (const char *) __attribute__((__noreturn__))`
- `void __throw_regex_error (regex\_constants::error\_type __ecode)`
- `void __throw_runtime_error (const char *) __attribute__((__noreturn__))`
- `void __throw_system_error (int) __attribute__((__noreturn__))`

- `void __throw_underflow_error (const char *) __attribute__((__noreturn__))`
- `template<typename _Ex > void __throw_with_nested (_Ex &&, const nested_exception *=0) __attribute__((__noreturn__))`
- `template<typename _Ex > void __throw_with_nested (_Ex &&,...) __attribute__((__noreturn__))`
- `template<typename _RandomAccessIterator > void __unguarded_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare > void __unguarded_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator > void __unguarded_linear_insert (_RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare > void __unguarded_linear_insert (_RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Tp > _RandomAccessIterator __unguarded_partition (_RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp &__pivot)`
- `template<typename _RandomAccessIterator, typename _Tp, typename _Compare > _RandomAccessIterator __unguarded_partition (_RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp &__pivot, _Compare __comp)`
- `template<typename _RandomAccessIterator > _RandomAccessIterator __unguarded_partition_pivot (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare > _RandomAccessIterator __unguarded_partition_pivot (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Pointer, typename _ForwardIterator > void __uninitialized_construct_buf (_Pointer __first, _Pointer __last, _ForwardIterator __seed)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator > _ForwardIterator __uninitialized_copy_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp > _ForwardIterator __uninitialized_copy_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, allocator<_Tp> &__a)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, typename _Allocator > _ForwardIterator __uninitialized_copy_move (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator > _ForwardIterator __uninitialized_copy_n (_InputIterator __first, _Size __n, _ForwardIterator __result, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Size, typename _ForwardIterator > _ForwardIterator __uninitialized_copy_n (_RandomAccessIterator __first, _Size __n, _ForwardIterator __result, random_access_iterator_tag)`
- `template<typename _ForwardIterator > void __uninitialized_default (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Allocator > void __uninitialized_default_a (_ForwardIterator __first, _ForwardIterator __last, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Tp > void __uninitialized_default_a (_ForwardIterator __first, _ForwardIterator __last, allocator<_Tp> &__a)`
- `template<typename _ForwardIterator, typename _Size > void __uninitialized_default_n (_ForwardIterator __first, _Size __n)`
- `template<typename _ForwardIterator, typename _Size, typename _Allocator > void __uninitialized_default_n_a (_ForwardIterator __first, _Size __n, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp > void __uninitialized_default_n_a (_ForwardIterator __first, _Size __n, allocator<_Tp> &__a)`
- `template<typename _ForwardIterator, typename _Tp, typename _Allocator > void __uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Tp, typename _Tp2 > void __uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x, allocator<_Tp2> &__a)`
- `template<typename _ForwardIterator, typename _Tp, typename _InputIterator, typename _Allocator > _ForwardIterator __uninitialized_fill_move (_ForwardIterator __result, _ForwardIterator __mid, const _Tp &__x, _InputIterator __first, _InputIterator __last, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Allocator > void __uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const _Tp &__x, _Allocator &__alloc)`



- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Tp2> void __uninitialized_fill_n_a (↵  
_ForwardIterator __first, _Size __n, const _Tp &__x, allocator<_Tp2> &)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator> _ForwardIterator __uninitialized_↵  
move_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, typename _Allocator> _ForwardIterator  
__uninitialized_move_copy (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Input↵  
Iterator2 __last2, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp, typename _Allocator> void __uninitialized_↵  
move_fill (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, const  
_Tp &__x, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator> _ForwardIterator __uninitialized_↵  
move_if_noexcept_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _OutputIterator> _OutputIterator __unique_copy (_ForwardIterator __first,  
_ForwardIterator __last, _OutputIterator __result, forward_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _OutputIterator> _OutputIterator __unique_copy (_InputIterator __first, ↵  
_InputIterator __last, _OutputIterator __result, input_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _ForwardIterator> _ForwardIterator __unique_copy (_InputIterator __first, ↵  
_InputIterator __last, _ForwardIterator __result, input_iterator_tag, forward_iterator_tag)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _BinaryPredicate> _OutputIterator __unique_copy  
(_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred,  
forward_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate> _OutputIterator __unique_copy (↵  
_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, input_↵  
iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate> _ForwardIterator __unique_copy  
(_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _BinaryPredicate __binary_pred, input_↵  
iterator_tag, forward_iterator_tag)`
- `template<typename _Bi_iter> const sub_match<_Bi_iter> &__unmatched_sub ()`
- `template<typename _Tp, typename _Alloc, typename... _Args> __uses_alloc_impl<_Tp, _Alloc, _Args...> __use_alloc  
(const _Alloc &__a)`
- `template<typename _Tp> void __valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b)`
- `template<typename _Tp> void __valarray_copy (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__↵  
restrict __b)`
- `template<typename _Tp> void __valarray_copy (const _Tp *__restrict __a, _Tp *__restrict __b, size_t __n,  
size_t __s)`
- `template<typename _Tp> void __valarray_copy (const _Tp *__restrict __src, size_t __n, size_t __s1, _Tp *__↵  
restrict __dst, size_t __s2)`
- `template<typename _Tp> void __valarray_copy (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp  
*__restrict __b, size_t __n)`
- `template<typename _Tp> void __valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b,  
const size_t *__restrict __i)`
- `template<typename _Tp> void __valarray_copy (const _Tp *__restrict __src, size_t __n, const size_t *__↵  
restrict __i, _Tp *__restrict __dst, const size_t *__restrict __j)`
- `template<typename _Tp> void __valarray_copy (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b)`
- `template<typename _Tp> void __valarray_copy (_Array<_Tp> __a, size_t __n, size_t __s, _Array<_Tp> __b)`
- `template<typename _Tp> void __valarray_copy (_Array<_Tp> __a, _Array<_Tp> __b, size_t __n, size_t __s)`
- `template<typename _Tp> void __valarray_copy (_Array<_Tp> __a, size_t __n, size_t __s1, _Array<_Tp> __b,  
size_t __s2)`
- `template<typename _Tp> void __valarray_copy (_Array<_Tp> __a, _Array<size_t> __i, _Array<_Tp> __b,  
size_t __n)`
- `template<typename _Tp> void __valarray_copy (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b, _Array<  
size_t> __i)`

- `template<typename _Tp> void __valarray_copy (_Array< _Tp> __src, size_t __n, _Array< size_t> __i, _Array< _Tp> __dst, _Array< size_t> __j)`
- `template<typename _Tp> void __valarray_copy_construct (const _Tp * __b, const _Tp * __e, _Tp * __restrict __o)`
- `template<typename _Tp> void __valarray_copy_construct (const _Tp * __restrict __a, size_t __n, size_t __s, _Tp * __restrict __o)`
- `template<typename _Tp> void __valarray_copy_construct (const _Tp * __restrict __a, const size_t * __restrict __i, _Tp * __restrict __o, size_t __n)`
- `template<typename _Tp> void __valarray_copy_construct (_Array< _Tp> __a, _Array< size_t> __i, _Array< _Tp> __b, size_t __n)`
- `template<typename _Tp> void __valarray_copy_construct (_Array< _Tp> __a, size_t __n, size_t __s, _Array< _Tp> __b)`
- `template<typename _Tp> void __valarray_default_construct (_Tp * __b, _Tp * __e)`
- `template<typename _Tp> void __valarray_destroy_elements (_Tp * __b, _Tp * __e)`
- `template<typename _Tp> void __valarray_fill (_Tp * __restrict __a, size_t __n, const _Tp & __t)`
- `template<typename _Tp> void __valarray_fill (_Tp * __restrict __a, size_t __n, size_t __s, const _Tp & __t)`
- `template<typename _Tp> void __valarray_fill (_Tp * __restrict __a, const size_t * __restrict __i, size_t __n, const _Tp & __t)`
- `template<typename _Tp> void __valarray_fill (_Array< _Tp> __a, size_t __n, const _Tp & __t)`
- `template<typename _Tp> void __valarray_fill (_Array< _Tp> __a, size_t __n, size_t __s, const _Tp & __t)`
- `template<typename _Tp> void __valarray_fill (_Array< _Tp> __a, _Array< size_t> __i, size_t __n, const _Tp & __t)`
- `template<typename _Tp> void __valarray_fill_construct (_Tp * __b, _Tp * __e, const _Tp __t)`
- `void * __valarray_get_memory (size_t __n)`
- `template<typename _Tp> _Tp * __restrict __valarray_get_storage (size_t __n)`
- `template<typename _Ta> _Ta::value_type __valarray_max (const _Ta & __a)`
- `template<typename _Ta> _Ta::value_type __valarray_min (const _Ta & __a)`
- `template<typename _Tp> _Tp __valarray_product (const _Tp * __f, const _Tp * __l)`
- `void __valarray_release_memory (void * __p)`
- `template<typename _Tp> _Tp __valarray_sum (const _Tp * __f, const _Tp * __l)`
- `template<typename _CharT> ostreambuf_iterator< _CharT> __write (ostreambuf_iterator< _CharT> __s, const _CharT * __ws, int __len)`
- `template<typename _CharT, typename _OutIter> _OutIter __write (_OutIter __s, const _CharT * __ws, int __len)`
- `template<typename _Tp> void __Array_augmented__bitwise_and (_Array< _Tp> __a, _Array< size_t> __i, _Array< _Tp> __b, size_t __n)`
- `template<typename _Tp> void __Array_augmented__bitwise_and (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b, _Array< bool> __m)`
- `template<typename _Tp> void __Array_augmented__bitwise_and (_Array< _Tp> __a, size_t __n, const _Tp & __t)`
- `template<typename _Tp> void __Array_augmented__bitwise_and (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b)`
- `template<typename _Tp, class _Dom> void __Array_augmented__bitwise_and (_Array< _Tp> __a, const Expr< _Dom, _Tp> & __e, size_t __n)`
- `template<typename _Tp> void __Array_augmented__bitwise_and (_Array< _Tp> __a, size_t __n, size_t __s, _Array< _Tp> __b)`
- `template<typename _Tp> void __Array_augmented__bitwise_and (_Array< _Tp> __a, _Array< _Tp> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom> void __Array_augmented__bitwise_and (_Array< _Tp> __a, size_t __s, const Expr< _Dom, _Tp> & __e, size_t __n)`
- `template<typename _Tp> void __Array_augmented__bitwise_and (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b, _Array< size_t> __i)`

- `template<typename _Tp, class _Dom> void _Array_augmented_bitwise_and (_Array<_Tp> __a, _Array<size_t> __i, const _Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented_bitwise_and (_Array<_Tp> __a, _Array<bool> __m, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp, class _Dom> void _Array_augmented_bitwise_and (_Array<_Tp> __a, _Array<bool> __m, const _Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented_bitwise_or (_Array<_Tp> __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp> void _Array_augmented_bitwise_or (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b)`
- `template<typename _Tp, class _Dom> void _Array_augmented_bitwise_or (_Array<_Tp> __a, const _Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented_bitwise_or (_Array<_Tp> __a, size_t __n, size_t __s, _Array<_Tp> __b)`
- `template<typename _Tp> void _Array_augmented_bitwise_or (_Array<_Tp> __a, _Array<_Tp> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom> void _Array_augmented_bitwise_or (_Array<_Tp> __a, size_t __s, const _Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented_bitwise_or (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b, _Array<size_t> __i)`
- `template<typename _Tp, class _Dom> void _Array_augmented_bitwise_or (_Array<_Tp> __a, _Array<size_t> __i, const _Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented_bitwise_or (_Array<_Tp> __a, _Array<bool> __m, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp, class _Dom> void _Array_augmented_bitwise_or (_Array<_Tp> __a, _Array<bool> __m, const _Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented_bitwise_or (_Array<_Tp> __a, _Array<size_t> __i, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp> void _Array_augmented_bitwise_or (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b, _Array<bool> __m)`
- `template<typename _Tp, class _Dom> void _Array_augmented_bitwise_xor (_Array<_Tp> __a, _Array<bool> __m, const _Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented_bitwise_xor (_Array<_Tp> __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp> void _Array_augmented_bitwise_xor (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b)`
- `template<typename _Tp, class _Dom> void _Array_augmented_bitwise_xor (_Array<_Tp> __a, const _Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented_bitwise_xor (_Array<_Tp> __a, size_t __n, size_t __s, _Array<_Tp> __b)`
- `template<typename _Tp> void _Array_augmented_bitwise_xor (_Array<_Tp> __a, _Array<_Tp> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom> void _Array_augmented_bitwise_xor (_Array<_Tp> __a, size_t __s, const _Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented_bitwise_xor (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b, _Array<size_t> __i)`
- `template<typename _Tp, class _Dom> void _Array_augmented_bitwise_xor (_Array<_Tp> __a, _Array<size_t> __i, const _Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented_bitwise_xor (_Array<_Tp> __a, _Array<bool> __m, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp> void _Array_augmented_bitwise_xor (_Array<_Tp> __a, _Array<size_t> __i, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp> void _Array_augmented_bitwise_xor (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b, _Array<bool> __m)`



- `template<typename _Tp> void _Array_augmented__divides (_Array<_Tp> __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp> void _Array_augmented__divides (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b)`
- `template<typename _Tp, class _Dom> void _Array_augmented__divides (_Array<_Tp> __a, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented__divides (_Array<_Tp> __a, size_t __n, size_t __s, _Array<_Tp> __b)`
- `template<typename _Tp> void _Array_augmented__divides (_Array<_Tp> __a, _Array<_Tp> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom> void _Array_augmented__divides (_Array<_Tp> __a, size_t __s, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented__divides (_Array<_Tp> __a, _Array<size_t> __i, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp> void _Array_augmented__divides (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b, _Array<size_t> __i)`
- `template<typename _Tp, class _Dom> void _Array_augmented__divides (_Array<_Tp> __a, _Array<size_t> __i, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented__divides (_Array<_Tp> __a, _Array<bool> __m, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp, class _Dom> void _Array_augmented__divides (_Array<_Tp> __a, _Array<bool> __m, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented__divides (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b, _Array<bool> __m)`
- `template<typename _Tp> void _Array_augmented__minus (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b, _Array<bool> __m)`
- `template<typename _Tp> void _Array_augmented__minus (_Array<_Tp> __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp> void _Array_augmented__minus (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b)`
- `template<typename _Tp> void _Array_augmented__minus (_Array<_Tp> __a, size_t __n, size_t __s, _Array<_Tp> __b)`
- `template<typename _Tp> void _Array_augmented__minus (_Array<_Tp> __a, _Array<_Tp> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom> void _Array_augmented__minus (_Array<_Tp> __a, size_t __s, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented__minus (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b, _Array<size_t> __i)`
- `template<typename _Tp, class _Dom> void _Array_augmented__minus (_Array<_Tp> __a, _Array<size_t> __i, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented__minus (_Array<_Tp> __a, _Array<bool> __m, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp, class _Dom> void _Array_augmented__minus (_Array<_Tp> __a, _Array<bool> __m, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp, class _Dom> void _Array_augmented__minus (_Array<_Tp> __a, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented__minus (_Array<_Tp> __a, _Array<size_t> __i, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp, class _Dom> void _Array_augmented__modulus (_Array<_Tp> __a, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented__modulus (_Array<_Tp> __a, _Array<size_t> __i, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp> void _Array_augmented__modulus (_Array<_Tp> __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp> void _Array_augmented__modulus (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b)`

- `template<typename _Tp> void _Array_augmented_modulus (_Array<_Tp> __a, size_t __n, size_t __s, _Array<_Tp> __b)`
- `template<typename _Tp> void _Array_augmented_modulus (_Array<_Tp> __a, _Array<_Tp> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom> void _Array_augmented_modulus (_Array<_Tp> __a, size_t __s, const _Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented_modulus (_Array<_Tp> __a, _Array<bool> __m, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp> void _Array_augmented_modulus (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b, _Array<bool> __m)`
- `template<typename _Tp, class _Dom> void _Array_augmented_modulus (_Array<_Tp> __a, _Array<bool> __m, const _Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented_modulus (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b, _Array<size_t> __i)`
- `template<typename _Tp, class _Dom> void _Array_augmented_modulus (_Array<_Tp> __a, _Array<size_t> __i, const _Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented_multiplies (_Array<_Tp> __a, _Array<size_t> __i, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp> void _Array_augmented_multiplies (_Array<_Tp> __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp> void _Array_augmented_multiplies (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b)`
- `template<typename _Tp, class _Dom> void _Array_augmented_multiplies (_Array<_Tp> __a, const _Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented_multiplies (_Array<_Tp> __a, size_t __n, size_t __s, _Array<_Tp> __b)`
- `template<typename _Tp> void _Array_augmented_multiplies (_Array<_Tp> __a, _Array<_Tp> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom> void _Array_augmented_multiplies (_Array<_Tp> __a, size_t __s, const _Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp, class _Dom> void _Array_augmented_multiplies (_Array<_Tp> __a, _Array<size_t> __i, const _Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented_multiplies (_Array<_Tp> __a, _Array<bool> __m, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp, class _Dom> void _Array_augmented_multiplies (_Array<_Tp> __a, _Array<bool> __m, const _Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented_multiplies (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b, _Array<size_t> __i)`
- `template<typename _Tp> void _Array_augmented_multiplies (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b, _Array<bool> __m)`
- `template<typename _Tp> void _Array_augmented_plus (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b, _Array<bool> __m)`
- `template<typename _Tp> void _Array_augmented_plus (_Array<_Tp> __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp> void _Array_augmented_plus (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b)`
- `template<typename _Tp> void _Array_augmented_plus (_Array<_Tp> __a, size_t __n, size_t __s, _Array<_Tp> __b)`
- `template<typename _Tp> void _Array_augmented_plus (_Array<_Tp> __a, _Array<_Tp> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom> void _Array_augmented_plus (_Array<_Tp> __a, size_t __s, const _Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented_plus (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b, _Array<size_t> __i)`
- `template<typename _Tp, class _Dom> void _Array_augmented_plus (_Array<_Tp> __a, _Array<size_t> __i, const _Expr<_Dom, _Tp> &__e, size_t __n)`

- `template<typename _Tp> void _Array_augmented_plus (_Array<_Tp> __a, _Array<bool> __m, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp, class _Dom> void _Array_augmented_plus (_Array<_Tp> __a, _Array<bool> __m, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented_plus (_Array<_Tp> __a, _Array<size_t> __i, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp, class _Dom> void _Array_augmented_plus (_Array<_Tp> __a, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented_shift_left (_Array<_Tp> __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp> void _Array_augmented_shift_left (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b)`
- `template<typename _Tp> void _Array_augmented_shift_left (_Array<_Tp> __a, size_t __n, size_t __s, _Array<_Tp> __b)`
- `template<typename _Tp> void _Array_augmented_shift_left (_Array<_Tp> __a, _Array<_Tp> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom> void _Array_augmented_shift_left (_Array<_Tp> __a, size_t __s, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented_shift_left (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b, _Array<size_t> __i)`
- `template<typename _Tp, class _Dom> void _Array_augmented_shift_left (_Array<_Tp> __a, _Array<size_t> __i, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented_shift_left (_Array<_Tp> __a, _Array<bool> __m, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp, class _Dom> void _Array_augmented_shift_left (_Array<_Tp> __a, _Array<bool> __m, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented_shift_left (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b, _Array<bool> __m)`
- `template<typename _Tp, class _Dom> void _Array_augmented_shift_left (_Array<_Tp> __a, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented_shift_left (_Array<_Tp> __a, _Array<size_t> __i, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp> void _Array_augmented_shift_right (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b, _Array<bool> __m)`
- `template<typename _Tp, class _Dom> void _Array_augmented_shift_right (_Array<_Tp> __a, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented_shift_right (_Array<_Tp> __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp> void _Array_augmented_shift_right (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b)`
- `template<typename _Tp> void _Array_augmented_shift_right (_Array<_Tp> __a, size_t __n, size_t __s, _Array<_Tp> __b)`
- `template<typename _Tp> void _Array_augmented_shift_right (_Array<_Tp> __a, _Array<_Tp> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom> void _Array_augmented_shift_right (_Array<_Tp> __a, size_t __s, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented_shift_right (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b, _Array<size_t> __i)`
- `template<typename _Tp, class _Dom> void _Array_augmented_shift_right (_Array<_Tp> __a, _Array<size_t> __i, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void _Array_augmented_shift_right (_Array<_Tp> __a, _Array<bool> __m, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp, class _Dom> void _Array_augmented_shift_right (_Array<_Tp> __a, _Array<bool> __m, const Expr<_Dom, _Tp> &__e, size_t __n)`

- `template<typename _Tp> void _Array_augmented_shift_right (_Array< _Tp> __a, _Array< size_t> __i, _Array< _Tp> __b, size_t __n)`
- `template<typename _T1, typename... _Args> void _Construct (_T1 *__p, _Args &&... __args)`
- `template<typename _Tp> void _Destroy (_Tp *__pointer)`
- `template<typename _ForwardIterator> void _Destroy (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Allocator> void _Destroy (_ForwardIterator __first, _ForwardIterator __last, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Tp> void _Destroy (_ForwardIterator __first, _ForwardIterator __last, allocator< _Tp> &)`
- `size_t Fnv_hash_bytes (const void *__ptr, size_t __len, size_t __seed)`
- `size_t Hash_bytes (const void *__ptr, size_t __len, size_t __seed)`
- `unsigned int Rb_tree_black_count (const _Rb_tree_node_base *__node, const _Rb_tree_node_base *__root) throw ()`
- `_Rb_tree_node_base * Rb_tree_decrement (_Rb_tree_node_base *__x) throw ()`
- `const _Rb_tree_node_base * Rb_tree_decrement (const _Rb_tree_node_base *__x) throw ()`
- `_Rb_tree_node_base * Rb_tree_increment (_Rb_tree_node_base *__x) throw ()`
- `const _Rb_tree_node_base * Rb_tree_increment (const _Rb_tree_node_base *__x) throw ()`
- `void Rb_tree_insert_and_rebalance (const bool __insert_left, _Rb_tree_node_base *__x, _Rb_tree_node_base *__p, _Rb_tree_node_base &__header) throw ()`
- `_Rb_tree_node_base * Rb_tree_rebalance_for_erase (_Rb_tree_node_base *const __z, _Rb_tree_node_base &__header) throw ()`
- `template<class _Dom> _Expr< _UnClos< _Abs, _Expr, _Dom>, typename _Dom::value_type> abs (const _Expr< _Dom, typename _Dom::value_type> &__e)`
- `template<typename _Tp> _Expr< _UnClos< _Abs, _ValArray, _Tp>, _Tp> abs (const valarray< _Tp> &__v)`
- `template<typename _InputIterator, typename _Tp> _Tp accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation> _Tp accumulate (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)`
- `template<class _Dom> _Expr< _UnClos< _Acos, _Expr, _Dom>, typename _Dom::value_type> acos (const _Expr< _Dom, typename _Dom::value_type> &__e)`
- `template<typename _Tp> _Expr< _UnClos< _Acos, _ValArray, _Tp>, _Tp> acos (const valarray< _Tp> &__v)`
- `template<typename _Tp> _Tp * addressof (_Tp &__r) noexcept`
- `template<typename _InputIterator, typename _OutputIterator> _OutputIterator adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation> _OutputIterator adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _Filter> _Filter adjacent_find (_Filter, _Filter)`
- `template<typename _Filter, typename _BinaryPredicate> _Filter adjacent_find (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _ForwardIterator> _ForwardIterator adjacent_find (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate> _ForwardIterator adjacent_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Distance> void advance (_InputIterator &__i, _Distance __n)`
- `template<typename _Iter, typename _Predicate> bool all_of (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate> bool all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Tp, typename _Alloc, typename... _Args> shared_ptr< _Tp> allocate_shared (const _Alloc &__a, _Args &&... __args)`
- `template<typename _Iter, typename _Predicate> bool any_of (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate> bool any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`

- `template<class _Dom> _Expr< _UnClos< _Asin, _Expr, _Dom>, typename _Dom::value_type> asin (const _Expr< _Dom, typename _Dom::value_type> &__e)`
- `template<typename _Tp> _Expr< _UnClos< _Asin, _ValArray, _Tp>, _Tp> asin (const valarray< _Tp> &__v)`
- `template<class _Dom> _Expr< _UnClos< _Atan, _Expr, _Dom>, typename _Dom::value_type> atan (const _Expr< _Dom, typename _Dom::value_type> &__e)`
- `template<typename _Tp> _Expr< _UnClos< _Atan, _ValArray, _Tp>, _Tp> atan (const valarray< _Tp> &__v)`
- `template<class _Dom1, class _Dom2> _Expr< _BinClos< _Atan2, _Expr, _Expr, _Dom1, _Dom2>, typename _Dom1::value_type> atan2 (const _Expr< _Dom1, typename _Dom1::value_type> &__e1, const _Expr< _Dom2, typename _Dom2::value_type> &__e2)`
- `template<class _Dom> _Expr< _BinClos< _Atan2, _Expr, _ValArray, _Dom, typename _Dom::value_type>, typename _Dom::value_type> atan2 (const _Expr< _Dom, typename _Dom::value_type> &__e, const valarray< typename _Dom::value_type> &__v)`
- `template<class _Dom> _Expr< _BinClos< _Atan2, _ValArray, _Expr, typename _Dom::value_type, _Dom>, typename _Dom::value_type> atan2 (const valarray< typename _Dom::valarray> &__v, const _Expr< _Dom, typename _Dom::value_type> &__e)`
- `template<class _Dom> _Expr< _BinClos< _Atan2, _Expr, _Constant, _Dom, typename _Dom::value_type>, typename _Dom::value_type> atan2 (const _Expr< _Dom, typename _Dom::value_type> &__e, const typename _Dom::value_type &__t)`
- `template<class _Dom> _Expr< _BinClos< _Atan2, _Constant, _Expr, typename _Dom::value_type, _Dom>, typename _Dom::value_type> atan2 (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type> &__e)`
- `template<typename _Tp> _Expr< _BinClos< _Atan2, _ValArray, _Constant, _Tp, _Tp>, _Tp> atan2 (const valarray< _Tp> &__v, const _Tp &__t)`
- `template<typename _Tp> _Expr< _BinClos< _Atan2, _Constant, _ValArray, _Tp, _Tp>, _Tp> atan2 (const _Tp &__t, const valarray< _Tp> &__v)`
- `template<typename _Tp> _Expr< _BinClos< _Atan2, _ValArray, _ValArray, _Tp, _Tp>, _Tp> atan2 (const valarray< _Tp> &__v, const valarray< _Tp> &__w)`
- `void atomic_signal_fence (memory\_order __m) noexcept`
- `void atomic_thread_fence (memory\_order __m) noexcept`
- `template<typename _Container> back\_insert\_iterator< _Container> back_inserter (_Container &__x)`
- `template<class _Container> auto begin (_Container &__cont) -> decltype(__cont.begin())`
- `template<class _Container> auto begin (const _Container &__cont) -> decltype(__cont.begin())`
- `template<class _Tp, size_t _Nm> _Tp * begin (_Tp(&__arr)[_Nm])`
- `template<typename _Filter, typename _Tp> bool binary_search (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare> bool binary_search (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _ForwardIterator, typename _Tp> bool binary\_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare> bool binary\_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _Operation, typename _Tp> binder1st< _Operation> bind1st (const _Operation &__fn, const _Tp &__x)`
- `template<typename _Operation, typename _Tp> binder2nd< _Operation> bind2nd (const _Operation &__fn, const _Tp &__x)`
- `ios\_base & boolalpha (ios\_base &__base)`
- `template<typename _Tp, typename _Tp1> shared\_ptr< _Tp> const_pointer_cast (const shared\_ptr< _Tp1> &__r) noexcept`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp> __shared_ptr< _Tp, _Lp> const_pointer_cast (const __shared_ptr< _Tp1, _Lp> &__r) noexcept`
- `template<typename _Iter, typename _OIter> _OIter copy (_Iter, _Iter, _OIter)`
- `template<typename _CharT> __gnu_cxx::__enable_if< __is_char< _CharT> >::__value, ostreambuf\_iterator< _CharT> >::__type copy (istreambuf\_iterator< _CharT> __first, istreambuf\_iterator< _CharT> __last, ostreambuf\_iterator< _CharT> __result)`

- `template<typename _Tp > \_Deque\_iterator< _Tp, _Tp &, _Tp * > copy (\_Deque\_iterator< _Tp, const _Tp &, const _Tp * >, \_Deque\_iterator< _Tp, const _Tp &, const _Tp * >, \_Deque\_iterator< _Tp, _Tp &, _Tp * >)`
- `template<typename _Tp > \_Deque\_iterator< _Tp, _Tp &, _Tp * > copy (\_Deque\_iterator< _Tp, _Tp &, _Tp * > \_\_first, \_Deque\_iterator< _Tp, _Tp &, _Tp * > \_\_last, \_Deque\_iterator< _Tp, _Tp &, _Tp * > \_\_result)`
- `template<typename _II, typename _OI > _OI copy (_II \_\_first, _II \_\_last, _OI \_\_result)`
- `template<typename _Blter1, typename _Blter2 > _Blter2 copy_backward (_Blter1, _Blter1, _Blter2)`
- `template<typename _Tp > \_Deque\_iterator< _Tp, _Tp &, _Tp * > copy_backward (\_Deque\_iterator< _Tp, const _Tp &, const _Tp * >, \_Deque\_iterator< _Tp, const _Tp &, const _Tp * >, \_Deque\_iterator< _Tp, _Tp &, _Tp * >)`
- `template<typename _Tp > \_Deque\_iterator< _Tp, _Tp &, _Tp * > copy_backward (\_Deque\_iterator< _Tp, _Tp &, _Tp * > \_\_first, \_Deque\_iterator< _Tp, _Tp &, _Tp * > \_\_last, \_Deque\_iterator< _Tp, _Tp &, _Tp * > \_\_result)`
- `template<typename _BI1, typename _BI2 > _BI2 copy_backward (_BI1 \_\_first, _BI1 \_\_last, _BI2 \_\_result)`
- `template<typename _Ex > exception_ptr copy_exception (_Ex \_\_ex) noexcept`
- `template<typename _IIter, typename _OIter, typename _Predicate > _OIter copy_if (_IIter, _IIter, _OIter, _Predicate)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate > _OutputIterator copy_if (_InputIterator \_\_first, _InputIterator \_\_last, _OutputIterator \_\_result, _Predicate \_\_pred)`
- `template<typename _IIter, typename _Size, typename _OIter > _OIter copy_n (_IIter, _Size, _OIter)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator > _OutputIterator copy_n (_InputIterator \_\_first, _Size \_\_n, _OutputIterator \_\_result)`
- `template<typename _Tp > _Expr< _UnClos< _Cos, _ValArray, _Tp >, _Tp > cos (const valarray< _Tp > &\_\_v)`
- `template<class _Dom > _Expr< _UnClos< _Cos, _Expr, _Dom >, typename _Dom::value_type > cos (const \_\_v < _Expr< _Dom, typename _Dom::value_type > &\_\_e)`
- `template<class _Dom > _Expr< _UnClos< _Cosh, _Expr, _Dom >, typename _Dom::value_type > cosh (const \_\_v < _Expr< _Dom, typename _Dom::value_type > &\_\_e)`
- `template<typename _Tp > _Expr< _UnClos< _Cosh, _ValArray, _Tp >, _Tp > cosh (const valarray< _Tp > &\_\_v)`
- `template<typename _IIter, typename _Tp > iterator_traits< _IIter >::difference_type count (_IIter, _IIter, const _Tp &)`
- `template<typename _InputIterator, typename _Tp > iterator_traits< _InputIterator >::difference_type count (_InputIterator \_\_first, _InputIterator \_\_last, const _Tp & \_\_value)`
- `template<typename _IIter, typename _Predicate > iterator_traits< _IIter >::difference_type count_if (_IIter, _IIter, \_\_pred Predicate)`
- `template<typename _InputIterator, typename _Predicate > iterator_traits< _InputIterator >::difference_type count_if (_InputIterator \_\_first, _InputIterator \_\_last, _Predicate \_\_pred)`
- `exception_ptr current_exception () noexcept`
- `ios\_base & dec (ios\_base & \_\_base)`
- `template<typename _InputIterator > iterator_traits< _InputIterator >::difference_type distance (_InputIterator \_\_first, _InputIterator \_\_last)`
- `template<typename _Tp, typename _Tp1 > shared\_ptr< _Tp > dynamic_pointer_cast (const shared\_ptr< _Tp1 > &\_\_r) noexcept`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp> shared\_ptr< _Tp, _Lp > dynamic_pointer_cast (const shared\_ptr< _Tp1, _Lp > &\_\_r) noexcept`
- `template<class _Container > auto end (_Container &\_\_cont) -> decltype(\_\_cont.end())`
- `template<class _Container > auto end (const _Container &\_\_cont) -> decltype(\_\_cont.end())`
- `template<class _Tp, size_t _Nm> _Tp * end (_Tp(&\_\_arr)[_Nm])`
- `template<typename _IIter1, typename _IIter2 > bool equal (_IIter1, _IIter1, _IIter2)`
- `template<typename _IIter1, typename _IIter2, typename _BinaryPredicate > bool equal (_IIter1 \_\_first1, _IIter1 \_\_last1, _IIter2 \_\_first2, _BinaryPredicate \_\_binary\_pred)`
- `template<typename _II1, typename _II2 > bool equal (_II1 \_\_first1, _II1 \_\_last1, _II2 \_\_first2)`
- `template<typename _Filter, typename _Tp > pair< _Filter, _Filter > equal_range (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare > pair< _Filter, _Filter > equal_range (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _ForwardIterator, typename _Tp > pair< _ForwardIterator, _ForwardIterator > equal_range (_ForwardIterator \_\_first, _ForwardIterator \_\_last, const _Tp & \_\_val)`

- `template<typename _ForwardIterator, typename _Tp, typename _Compare> pair<_ForwardIterator, _ForwardIterator> equal\_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<class _Dom> _Expr<_UnClos<_Exp, _Expr, _Dom>, typename _Dom::value_type> exp (const _Expr<_Dom, typename _Dom::value_type> &__e)`
- `template<typename _Tp> _Expr<_UnClos<_Exp, _ValArray, _Tp>, _Tp> exp (const valarray<_Tp> &__v)`
- `template<typename _Filter, typename _Tp> void fill (_Filter, _Filter, const _Tp &)`
- `template<typename _Tp> void fill (const Deque\_iterator<_Tp, _Tp &, _Tp * > &, const Deque\_iterator<_Tp, _Tp &, _Tp * > &, const _Tp &)`
- `void fill (_Bit_iterator __first, _Bit_iterator __last, const bool &__x)`
- `template<typename _ForwardIterator, typename _Tp> void fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _OIter, typename _Size, typename _Tp> _OIter fill_n (_OIter, _Size, const _Tp &)`
- `template<typename _OI, typename _Size, typename _Tp> _OI fill_n (_OI __first, _Size __n, const _Tp &__value)`
- `template<typename _CharT> gnu\_cxx::enable\_if<_is_char<_CharT>::value, istreambuf\_iterator<_CharT> >::type find (istreambuf\_iterator<_CharT> __first, istreambuf\_iterator<_CharT> __last, const _CharT &__val)`
- `template<typename _IOIter, typename _Tp> _IOIter find (_IOIter, _IOIter, const _Tp &)`
- `template<typename _InputIterator, typename _Tp> _InputIterator find (_InputIterator __first, _InputIterator __last, const _Tp &__val)`
- `template<typename _Filter1, typename _Filter2> _Filter1 find_end (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate> _Filter1 find_end (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2> _ForwardIterator1 find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate> _ForwardIterator1 find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)`
- `template<typename _Filter1, typename _Filter2> _Filter1 find_first_of (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate> _Filter1 find_first_of (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _InputIterator, typename _ForwardIterator> _InputIterator find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate> _InputIterator find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)`
- `template<typename _IOIter, typename _Predicate> _IOIter find_if (_IOIter, _IOIter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate> _InputIterator find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _IOIter, typename _Predicate> _IOIter find_if_not (_IOIter, _IOIter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate> _InputIterator find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `ios\_base & fixed (ios\_base &__base)`
- `template<typename _IOIter, typename _Funct> _Funct for_each (_IOIter, _IOIter, _Funct)`
- `template<typename _InputIterator, typename _Function> _Function for_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `template<typename _Tp> constexpr _Tp && forward (typename std::remove_reference<_Tp>::type &__t) noexcept`
- `template<typename _Tp> constexpr _Tp && forward (typename std::remove_reference<_Tp>::type &&__t) noexcept`
- `template<typename _Container> front\_insert\_iterator<_Container> front\_inserter (_Container &__x)`
- `template<typename _Filter, typename _Generator> void generate (_Filter, _Filter, _Generator)`
- `template<typename _ForwardIterator, typename _Generator> void generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __gen)`



- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator> _RealType generate\_canonical (↵  
_UniformRandomNumberGenerator &__g)`
- `template<typename _OIter, typename _Size, typename _Generator> _OIter generate\_n (_OIter, _Size, _Generator)`
- `template<typename _OutputIterator, typename _Size, typename _Generator> _OutputIterator generate\_n (_OutputIterator  
__first, _Size __n, _Generator __gen)`
- `template<typename _Del, typename _Tp, _Lock_policy _Lp> _Del * get\_deleter (const __shared_ptr< _Tp, _Lp > &__p)  
noexcept`
- `template<typename _Tp> pair< _Tp *, ptrdiff_t > get\_temporary\_buffer (ptrdiff_t __len) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> basic↵  
_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, \_\_gnu\_cxx::\_\_versa\_string<  
_CharT, _Traits, _Alloc, _Base > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> basic↵  
_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, \_\_gnu\_cxx::\_\_versa\_string<  
_CharT, _Traits, _Alloc, _Base > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc> basic_istream< _CharT, _Traits > & getline (basic↵  
istream< _CharT, _Traits > &__is, basic\_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc> basic_istream< _CharT, _Traits > & getline (basic↵  
istream< _CharT, _Traits > &__is, basic\_string< _CharT, _Traits, _Alloc > &__str)`
- `template<> basic_istream< char > & getline (basic_istream< char > &__in, basic\_string< char > &__str, char  
__delim)`
- `template<> basic_istream< wchar_t > & getline (basic_istream< wchar_t > &__in, basic\_string< wchar_t >  
&__str, wchar_t __delim)`
- `template<typename _Facet> bool has\_facet (const locale &) throw ()`
- `ios\_base & hex (ios\_base &__base)`
- `template<typename _Iter1, typename _Iter2> bool includes (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare> bool includes (_Iter1, _Iter1, _Iter2, _Iter2, ↵  
_Compare)`
- `template<typename _InputIterator1, typename _InputIterator2> bool includes (_InputIterator1 __first1, _InputIterator1 ↵  
__last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare> bool includes (_InputIterator1 __first1,  
_InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp> _Tp inner\_product (_InputIterator1 __first1,  
_InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2  
> _Tp inner\_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, ↵  
_BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)`
- `template<typename _BIter> void inplace\_merge (_BIter, _BIter, _BIter)`
- `template<typename _BIter, typename _Compare> void inplace\_merge (_BIter, _BIter, _BIter, _Compare)`
- `template<typename _BidirectionalIterator> void inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator ↵  
__middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare> void inplace\_merge (_BidirectionalIterator __first, ↵  
_BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _Container, typename _Iterator> insert\_iterator< _Container > inserter (_Container &__x, _Iterator  
__i)`
- `ios\_base & internal (ios\_base &__base)`
- `template<typename _ForwardIterator, typename _Tp> void iota (_ForwardIterator __first, _ForwardIterator __last, _Tp  
__value)`
- `template<typename _RAIter> bool is\_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare> bool is\_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator> bool is\_heap (_RandomAccessIterator __first, _RandomAccessIterator ↵  
__last)`



- `template<typename _RandomAccessIterator, typename _Compare> bool is\_heap (_RandomAccessIterator __first, _↵  
RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RAIter> _RAIter is\_heap\_until (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare> _RAIter is\_heap\_until (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator> _RandomAccessIterator is\_heap\_until (_RandomAccessIterator __first,  
_RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare> _RandomAccessIterator is\_heap\_until (_Random↵  
AccessIterator __first, RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Iter, typename _Predicate> bool is\_partitioned (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate> bool is\_partitioned (_InputIterator __first, _InputIterator __last,  
_Predicate __pred)`
- `template<typename _Filter1, typename _Filter2> bool is\_permutation (_Filter1, _Filter1, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate> bool is\_permutation (_Filter1, _Filter1, _Filter2,  
_BinaryPredicate)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2> bool is\_permutation (_ForwardIterator1 __first1, _↵  
ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate> bool is\_permutation (_↵  
ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)`
- `template<typename _Filter> bool is\_sorted (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare> bool is\_sorted (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator> bool is\_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare> bool is\_sorted (_ForwardIterator __first, _ForwardIterator _↵  
__last, _Compare __comp)`
- `template<typename _Filter> _Filter is\_sorted\_until (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare> _Filter is\_sorted\_until (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator> _ForwardIterator is\_sorted\_until (_ForwardIterator __first, _ForwardIterator _↵  
__last)`
- `template<typename _ForwardIterator, typename _Compare> _ForwardIterator is\_sorted\_until (_ForwardIterator __first, ↵  
_ForwardIterator __last, _Compare __comp)`
- `template<typename _CharT> bool isalnum (_CharT __c, const locale &__loc)`
- `template<typename _CharT> bool isalpha (_CharT __c, const locale &__loc)`
- `template<typename _CharT> bool iscntrl (_CharT __c, const locale &__loc)`
- `template<typename _CharT> bool isdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT> bool isgraph (_CharT __c, const locale &__loc)`
- `template<typename _CharT> bool islower (_CharT __c, const locale &__loc)`
- `template<typename _CharT> bool isprint (_CharT __c, const locale &__loc)`
- `template<typename _CharT> bool ispunct (_CharT __c, const locale &__loc)`
- `template<typename _CharT> bool isspace (_CharT __c, const locale &__loc)`
- `template<typename _CharT> bool isupper (_CharT __c, const locale &__loc)`
- `template<typename _CharT> bool isxdigit (_CharT __c, const locale &__loc)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2> void iter\_swap (_ForwardIterator1 __a, _Forward↵  
Iterator2 __b)`
- `template<typename _Filter1, typename _Filter2> void iter\_swap (_Filter1, _Filter2)`
- `template<typename _Tp> _Tp kill\_dependency (_Tp __y) noexcept`
- `ios\_base & left (ios\_base &__base)`
- `template<typename _Iter1, typename _Iter2> bool lexicographical\_compare (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare> bool lexicographical\_compare (_Iter1, _Iter1, _I↵  
Iter2, _Iter2, _Compare)`
- `template<typename _II1, typename _II2> bool lexicographical\_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2  
__last2)`
- `template<typename _II1, typename _II2, typename _Compare> bool lexicographical\_compare (_II1 __first1, _II1 __last1,  
_II2 __first2, _II2 __last2, _Compare __comp)`

- `template<class _Dom > _Expr< _UnClos< _Log, _Expr, _Dom >, typename _Dom::value_type > log (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp > _Expr< _UnClos< _Log, _ValArray, _Tp >, _Tp > log (const valarray< _Tp > &__v)`
- `template<class _Dom > _Expr< _UnClos< _Log10, _Expr, _Dom >, typename _Dom::value_type > log10 (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp > _Expr< _UnClos< _Log10, _ValArray, _Tp >, _Tp > log10 (const valarray< _Tp > &__v)`
- `template<typename _Filter, typename _Tp > _Filter lower_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare > _Filter lower_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _ForwardIterator, typename _Tp > _ForwardIterator lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare > _ForwardIterator lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _Ex > exception_ptr make_exception_ptr (_Ex __ex) noexcept`
- `template<typename _RAIter > void make_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare > void make_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator > void make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare > void make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Iterator > move_iterator< _Iterator > make_move_iterator (_Iterator __i)`
- `template<class _T1, class _T2 > constexpr pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_strip< _T2 >::__type > make_pair (_T1 &&__x, _T2 &&__y)`
- `template<typename _Tp, typename... _Args> shared_ptr< _Tp > make_shared (_Args &&...__args)`
- `template<typename _Tp > const _Tp & max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare > const _Tp & max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp > _Tp max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare > _Tp max (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter > _Filter max_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare > _Filter max_element (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator > _ForwardIterator max_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare > _ForwardIterator max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Ret, typename _Tp > mem_fun_t< _Ret, _Tp > mem_fun (_Ret(_Tp::* __f)())`
- `template<typename _Ret, typename _Tp > const_mem_fun_t< _Ret, _Tp > mem_fun (_Ret(_Tp::* __f)() const)`
- `template<typename _Ret, typename _Tp, typename _Arg > mem_fun1_t< _Ret, _Tp, _Arg > mem_fun (_Ret(_Tp::* __f)(_Arg))`
- `template<typename _Ret, typename _Tp, typename _Arg > const_mem_fun1_t< _Ret, _Tp, _Arg > mem_fun (_Ret(_Tp::* __f)(_Arg) const)`
- `template<typename _Ret, typename _Tp > mem_fun_ref_t< _Ret, _Tp > mem_fun_ref (_Ret(_Tp::* __f)())`
- `template<typename _Ret, typename _Tp > const_mem_fun_ref_t< _Ret, _Tp > mem_fun_ref (_Ret(_Tp::* __f)() const)`
- `template<typename _Ret, typename _Tp, typename _Arg > mem_fun1_ref_t< _Ret, _Tp, _Arg > mem_fun_ref (_Ret(_Tp::* __f)(_Arg))`
- `template<typename _Ret, typename _Tp, typename _Arg > const_mem_fun1_ref_t< _Ret, _Tp, _Arg > mem_fun_ref (_Ret(_Tp::* __f)(_Arg) const)`
- `template<typename _Iter1, typename _Iter2, typename _OIter > _OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare > _OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator > _OutputIterator merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`

- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare> _OutputIterator merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _↵  
_OutputIterator __result, _Compare __comp)`
- `template<typename _Tp> const _Tp & min (const _Tp & __a, const _Tp & __b)`
- `template<typename _Tp, typename _Compare> const _Tp & min (const _Tp & __a, const _Tp & __b, _Compare __comp)`
- `template<typename _Tp> _Tp min (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare> _Tp min (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter> _Filter min\_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare> _Filter min\_element (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator> _ForwardIterator min\_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare> _ForwardIterator min\_element (_ForwardIterator __first, _↵  
_ForwardIterator __last, _Compare __comp)`
- `template<typename _Tp> pair< const _Tp &, const _Tp &> minmax (const _Tp & __a, const _Tp & __b)`
- `template<typename _Tp, typename _Compare> pair< const _Tp &, const _Tp &> minmax (const _Tp & __a, const _Tp  
& __b, _Compare __comp)`
- `template<typename _Tp> pair< _Tp, _Tp> minmax (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare> pair< _Tp, _Tp> minmax (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter> pair< _Filter, _Filter> minmax\_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare> pair< _Filter, _Filter> minmax\_element (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator> pair< _ForwardIterator, _ForwardIterator> minmax\_element (_ForwardIterator  
__first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare> pair< _ForwardIterator, _ForwardIterator> minmax\_element  
(_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2> pair< _Iter1, _Iter2> mismatch (_Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate> pair< _Iter1, _Iter2> mismatch (_Iter1, _Iter1,  
_Iter2, _BinaryPredicate)`
- `template<typename _InputIterator1, typename _InputIterator2> pair< _InputIterator1, _InputIterator2> mismatch (_Input↵  
_Iterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate> pair< _InputIterator1, _Input↵  
_Iterator2> mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate  
__binary_pred)`
- `template<typename _Tp> constexpr std::remove_reference< _Tp >::type && move (_Tp && __t) noexcept`
- `template<typename _Tp> Deque\_iterator< _Tp, _Tp &, _Tp * > move (_Deque_iterator< _Tp, const _Tp &, const  
_Tp * >, _Deque_iterator< _Tp, const _Tp &, const _Tp * >, _Deque_iterator< _Tp, _Tp &, _Tp * >)`
- `template<typename _Tp> Deque\_iterator< _Tp, _Tp &, _Tp * > move (_Deque_iterator< _Tp, _Tp &, _Tp * >  
__first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _II, typename _OI> _OI move (_II __first, _II __last, _OI __result)`
- `template<typename _Tp> Deque\_iterator< _Tp, _Tp &, _Tp * > move\_backward (_Deque_iterator< _Tp, const  
_Tp &, const _Tp * >, _Deque_iterator< _Tp, const _Tp &, const _Tp * >, _Deque_iterator< _Tp, _Tp &, _Tp *  
>)`
- `template<typename _Tp> Deque\_iterator< _Tp, _Tp &, _Tp * > move\_backward (_Deque_iterator< _Tp, _Tp &,  
_Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _BI1, typename _BI2> _BI2 move\_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _Tp> constexpr conditional< __move_if_noexcept_cond< _Tp >::value, const _Tp &, _Tp &&  
>::type move\_if\_noexcept (_Tp & __x) noexcept`
- `template<typename _ForwardIterator> _ForwardIterator next (_ForwardIterator __x, typename iterator_traits< _↵  
_ForwardIterator>::difference_type __n=1)`
- `template<typename _Blter> bool next\_permutation (_Blter, _Blter)`
- `template<typename _Blter, typename _Compare> bool next\_permutation (_Blter, _Blter, _Compare)`
- `template<typename _BidirectionalIterator> bool next\_permutation (_BidirectionalIterator __first, _BidirectionalIterator ↵  
__last)`

- `template<typename _BidirectionalIterator, typename _Compare> bool next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `ios_base & noboolalpha (ios_base & __base)`
- `template<typename _Iter, typename _Predicate> bool none_of (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate> bool none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `ios_base & noshowbase (ios_base & __base)`
- `ios_base & noshowpoint (ios_base & __base)`
- `ios_base & noshowpos (ios_base & __base)`
- `ios_base & noskipws (ios_base & __base)`
- `template<typename _Predicate> unary_negate< _Predicate> not1 (const _Predicate & __pred)`
- `template<typename _Predicate> binary_negate< _Predicate> not2 (const _Predicate & __pred)`
- `ios_base & nounitbuf (ios_base & __base)`
- `ios_base & nouppercase (ios_base & __base)`
- `template<typename _RAIter> void nth_element (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare> void nth_element (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator> void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare> void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp)`
- `ios_base & oct (ios_base & __base)`
- `template<class _Tp, class _CharT, class _Traits, class _Dist> bool operator!= (const istream_iterator< _Tp, _CharT, _Traits, _Dist> & __x, const istream_iterator< _Tp, _CharT, _Traits, _Dist> & __y)`
- `template<typename _T1, typename _T2> bool operator!= (const allocator< _T1> & __x, const allocator< _T2> & __y)`
- `template<typename _Tp> bool operator!= (const allocator< _Tp> & __x, const allocator< _Tp> & __y)`
- `template<typename _CharT, typename _Traits> bool operator!= (const istreambuf_iterator< _CharT, _Traits> & __a, const istreambuf_iterator< _CharT, _Traits> & __b)`
- `template<typename _StateT> bool operator!= (const fpos< _StateT> & __lhs, const fpos< _StateT> & __rhs)`
- `template<class _T1, class _T2> constexpr bool operator!= (const pair< _T1, _T2> & __x, const pair< _T1, _T2> & __y)`
- `template<typename _Tp, typename _Ref, typename _Ptr> bool operator!= (const _Deque_iterator< _Tp, _Ref, _Ptr> & __x, const _Deque_iterator< _Tp, _Ref, _Ptr> & __y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR> bool operator!= (const _Deque_iterator< _Tp, _RefL, _PtrL> & __x, const _Deque_iterator< _Tp, _RefR, _PtrR> & __y)`
- `template<typename _Tp> bool operator!= (const _Fwd_list_iterator< _Tp> & __x, const _Fwd_list_const_iterator< _Tp> & __y)`
- `template<typename _Tp, typename _Seq> bool operator!= (const stack< _Tp, _Seq> & __x, const stack< _Tp, _Seq> & __y)`
- `template<typename _Val> bool operator!= (const _List_iterator< _Val> & __x, const _List_const_iterator< _Val> & __y)`
- `template<typename _Tp, typename _Seq> bool operator!= (const queue< _Tp, _Seq> & __x, const queue< _Tp, _Seq> & __y)`
- `template<typename _Iterator> bool operator!= (const reverse_iterator< _Iterator> & __x, const reverse_iterator< _Iterator> & __y)`
- `template<typename _Val> bool operator!= (const _Rb_tree_iterator< _Val> & __x, const _Rb_tree_const_iterator< _Val> & __y)`
- `template<typename _Tp1, typename _Tp2> bool operator!= (const shared_ptr< _Tp1> & __a, const shared_ptr< _Tp2> & __b) noexcept`
- `template<typename _Tp> bool operator!= (const shared_ptr< _Tp> & __a, nullptr_t) noexcept`
- `template<typename _IteratorL, typename _IteratorR> bool operator!= (const reverse_iterator< _IteratorL> & __x, const reverse_iterator< _IteratorR> & __y)`
- `template<typename _Tp> bool operator!= (nullptr_t, const shared_ptr< _Tp> & __a) noexcept`

- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> bool operator!= (const std::linear\_congruential\_engine< _UIntType, __a, __c, __m > &__lhs, const std::linear\_congruential\_engine< _UIntType, __a, __c, __m > &__rhs)`
- `template<class _Dom1, class _Dom2> _Expr< _BinClos< __not_equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __not_equal_to, typename _Dom1::value_type >::result_type > operator!= (const _Expr< __not_equal_to, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom> _Expr< _BinClos< __not_equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __not_equal_to, typename _Dom::value_type >::result_type > operator!= (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom> _Expr< _BinClos< __not_equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __not_equal_to, typename _Dom::value_type >::result_type > operator!= (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom> _Expr< _BinClos< __not_equal_to, _Constant, _Expr, typename _Dom::value_type, __not_equal_to, typename _Dom::value_type >::result_type > operator!= (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom> _Expr< _BinClos< __not_equal_to, _ValArray, _Expr, typename _Dom::value_type, __not_equal_to, typename _Dom::value_type >::result_type > operator!= (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep> bool operator!= (const unique\_ptr< _Tp, _Dp > &__x, const unique\_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp> bool operator!= (const unique\_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp> bool operator!= (nullptr_t, const unique\_ptr< _Tp, _Dp > &__x)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, __not_equal_to, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> bool operator!= (const std::mersenne\_twister\_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__lhs, const std::mersenne\_twister\_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Alloc> bool operator!= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc> bool operator!= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r> bool operator!= (const std::subtract\_with\_carry\_engine< _UIntType, __w, __s, __r > &__lhs, const std::subtract\_with\_carry\_engine< _UIntType, __w, __s, __r > &__rhs)`
- `template<typename _Bilter> bool operator!= (const sub\_match< _Bilter > &__lhs, const sub\_match< _Bilter > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc> bool operator!= (const \_Rb\_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const \_Rb\_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> bool operator!= (const \_\_sub\_match\_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub\_match< _Bi_iter > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool operator!= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> bool operator!= (const sub\_match< _Bi_iter > &__lhs, const \_\_sub\_match\_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR> bool operator!= (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp> bool operator!= (const \_\_shared\_ptr< _Tp1, _Lp > &__a, const \_\_shared\_ptr< _Tp2, _Lp > &__b)`
- `template<typename _Iterator> bool operator!= (const move\_iterator< _Iterator > &__x, const move\_iterator< _Iterator > &__y)`
- `template<typename _Tp, _Lock_policy _Lp> bool operator!= (const \_\_shared\_ptr< _Tp, _Lp > &__a, nullptr_t)`



- `template<typename _Tp, _Lock_policy _Lp> bool operator!= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r> bool operator!= (const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__lhs, const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__rhs)`
- `template<typename _Bi_iter> bool operator!= (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter> bool operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const * __rhs)`
- `template<typename _Bi_iter> bool operator!= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> bool operator!= (const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__lhs, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__rhs)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc> bool operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc> bool operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Bi_iter> bool operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Tp, typename _Alloc> bool operator!= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc> bool operator!= (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc> bool operator!= (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc> bool operator!= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _RandomNumberEngine, size_t __k> bool operator!= (const std::shuffle_order_engine< _RandomNumberEngine, __k > &__lhs, const std::shuffle_order_engine< _RandomNumberEngine, __k > &__rhs)`
- `template<typename _Tp, typename _Alloc> bool operator!= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _IntType> bool operator!= (const std::uniform_int_distribution< _IntType > &__d1, const std::uniform_int_distribution< _IntType > &__d2)`
- `template<typename _Bi_iter, class _Alloc> bool operator!= (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)`
- `template<typename _Tp, typename _Alloc> bool operator!= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _IntType> bool operator!= (const std::uniform_real_distribution< _IntType > &__d1, const std::uniform_real_distribution< _IntType > &__d2)`
- `template<typename _RealType> bool operator!= (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _RealType > &__d2)`
- `template<typename _RealType> bool operator!= (const std::lognormal_distribution< _RealType > &__d1, const std::lognormal_distribution< _RealType > &__d2)`
- `template<typename _CharT, typename _Traits, typename _Alloc> bool operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc> bool operator!= (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc> bool operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT * __rhs)`
- `template<typename _RealType> bool operator!= (const std::gamma_distribution< _RealType > &__d1, const std::gamma_distribution< _RealType > &__d2)`

- `template<typename _RealType> bool operator!= (const std::chi_squared_distribution< _RealType> &__d1, const std::chi_squared_distribution< _RealType> &__d2)`
- `template<typename _RealType> bool operator!= (const std::cauchy_distribution< _RealType> &__d1, const std::cauchy_distribution< _RealType> &__d2)`
- `template<typename _RealType> bool operator!= (const std::fisher_f_distribution< _RealType> &__d1, const std::fisher_f_distribution< _RealType> &__d2)`
- `template<typename _RealType> bool operator!= (const std::student_t_distribution< _RealType> &__d1, const std::student_t_distribution< _RealType> &__d2)`
- `bool operator!= (const std::bernoulli_distribution &__d1, const std::bernoulli_distribution &__d2)`
- `template<typename _IntType> bool operator!= (const std::binomial_distribution< _IntType> &__d1, const std::binomial_distribution< _IntType> &__d2)`
- `template<typename _IntType> bool operator!= (const std::geometric_distribution< _IntType> &__d1, const std::geometric_distribution< _IntType> &__d2)`
- `template<typename _IntType> bool operator!= (const std::negative_binomial_distribution< _IntType> &__d1, const std::negative_binomial_distribution< _IntType> &__d2)`
- `template<typename _IntType> bool operator!= (const std::poisson_distribution< _IntType> &__d1, const std::poisson_distribution< _IntType> &__d2)`
- `template<typename _RealType> bool operator!= (const std::exponential_distribution< _RealType> &__d1, const std::exponential_distribution< _RealType> &__d2)`
- `template<typename _RealType> bool operator!= (const std::weibull_distribution< _RealType> &__d1, const std::weibull_distribution< _RealType> &__d2)`
- `template<typename _RealType> bool operator!= (const std::extreme_value_distribution< _RealType> &__d1, const std::extreme_value_distribution< _RealType> &__d2)`
- `template<typename _IntType> bool operator!= (const std::discrete_distribution< _IntType> &__d1, const std::discrete_distribution< _IntType> &__d2)`
- `template<typename _RealType> bool operator!= (const std::piecewise_constant_distribution< _RealType> &__d1, const std::piecewise_constant_distribution< _RealType> &__d2)`
- `template<typename _RealType> bool operator!= (const std::piecewise_linear_distribution< _RealType> &__d1, const std::piecewise_linear_distribution< _RealType> &__d2)`
- `template<class _Dom> _Expr< _BinClos< __modulus, _Expr, _ValArray, _Dom, typename _Dom::value_type>, typename __fun< __modulus, typename _Dom::value_type>::result_type> operator% (const _Expr< _Dom, typename _Dom::value_type> &__e, const valarray< typename _Dom::value_type> &__v)`
- `template<class _Dom> _Expr< _BinClos< __modulus, _Expr, _Constant, _Dom, typename _Dom::value_type>, typename __fun< __modulus, typename _Dom::value_type>::result_type> operator% (const _Expr< _Dom, typename _Dom::value_type> &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2> _Expr< _BinClos< __modulus, _Expr, _Expr, _Dom1, _Dom2>, typename __fun< __modulus, typename _Dom1::value_type>::result_type> operator% (const _Expr< _Dom1, typename _Dom1::value_type> &__v, const _Expr< _Dom2, typename _Dom2::value_type> &__w)`
- `template<class _Dom> _Expr< _BinClos< __modulus, _Constant, _Expr, typename _Dom::value_type, _Dom>, typename __fun< __modulus, typename _Dom::value_type>::result_type> operator% (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type> &__v)`
- `template<class _Dom> _Expr< _BinClos< __modulus, _ValArray, _Expr, typename _Dom::value_type, _Dom>, typename __fun< __modulus, typename _Dom::value_type>::result_type> operator% (const valarray< typename _Dom::value_type> &__v, const _Expr< _Dom, typename _Dom::value_type> &__e)`
- `constexpr _los_Fmtflags operator& (_los_Fmtflags __a, _los_Fmtflags __b)`
- `constexpr memory_order operator& (memory_order __m, __memory_order_modifier __mod)`
- `constexpr _los_Openmode operator& (_los_Openmode __a, _los_Openmode __b)`
- `constexpr _los_losestate operator& (_los_losestate __a, _los_losestate __b)`
- `template<class _Dom1, class _Dom2> _Expr< _BinClos< __bitwise_and, _Expr, _Expr, _Dom1, _Dom2>, typename __fun< __bitwise_and, typename _Dom1::value_type>::result_type> operator& (const _Expr< _Dom1, typename _Dom1::value_type> &__v, const _Expr< _Dom2, typename _Dom2::value_type> &__w)`

- `template<class _Dom > _Expr< _BinClos< __bitwise_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __bitwise_and, typename _Dom::value_type >::result_type > operator& (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom > _Expr< _BinClos< __bitwise_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __bitwise_and, typename _Dom::value_type >::result_type > operator& (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom > _Expr< _BinClos< __bitwise_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __bitwise_and, typename _Dom::value_type >::result_type > operator& (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom > _Expr< _BinClos< __bitwise_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __bitwise_and, typename _Dom::value_type >::result_type > operator& (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom > _Expr< _BinClos< __logical_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __logical_and, typename _Dom::value_type >::result_type > operator&& (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2 > _Expr< _BinClos< __logical_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __logical_and, typename _Dom1::value_type >::result_type > operator&& (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom > _Expr< _BinClos< __logical_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __logical_and, typename _Dom::value_type >::result_type > operator&& (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom > _Expr< _BinClos< __logical_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __logical_and, typename _Dom::value_type >::result_type > operator&& (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom > _Expr< _BinClos< __logical_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __logical_and, typename _Dom::value_type >::result_type > operator&& (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `const _ios_Fmtflags & operator&= ( _ios_Fmtflags &__a, _ios_Fmtflags __b)`
- `const _ios_Openmode & operator&= ( _ios_Openmode &__a, _ios_Openmode __b)`
- `const _ios_ostate & operator&= ( _ios_ostate &__a, _ios_ostate __b)`
- `template<class _Dom1, class _Dom2 > _Expr< _BinClos< __multiplies, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __multiplies, typename _Dom1::value_type >::result_type > operator* (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom > _Expr< _BinClos< __multiplies, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __multiplies, typename _Dom::value_type >::result_type > operator* (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom > _Expr< _BinClos< __multiplies, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __multiplies, typename _Dom::value_type >::result_type > operator* (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom > _Expr< _BinClos< __multiplies, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __multiplies, typename _Dom::value_type >::result_type > operator* (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom > _Expr< _BinClos< __multiplies, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __multiplies, typename _Dom::value_type >::result_type > operator* (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `_Bit_iterator operator+ (ptrdiff_t __n, const _Bit_iterator &__x)`
- `template<typename _Iterator > reverse_iterator< _Iterator > operator+ (typename reverse_iterator< _Iterator >::difference_type __n, const reverse_iterator< _Iterator > &__x)`
- `template<typename _Tp, typename _Ref, typename _Ptr > Deque_iterator< _Tp, _Ref, _Ptr > operator+ (ptrdiff_t __n, const Deque_iterator< _Tp, _Ref, _Ptr > &__x)`
- `_Bit_const_iterator operator+ (ptrdiff_t __n, const _Bit_const_iterator &__x)`



- `template<class _Dom > _Expr< _BinClos< __plus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __plus, typename _Dom::value_type >::result_type > operator+ (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom > _Expr< _BinClos< __plus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __plus, typename _Dom::value_type >::result_type > operator+ (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2 > _Expr< _BinClos< __plus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __plus, typename _Dom1::value_type >::result_type > operator+ (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom > _Expr< _BinClos< __plus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __plus, typename _Dom::value_type >::result_type > operator+ (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom > _Expr< _BinClos< __plus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __plus, typename _Dom::value_type >::result_type > operator+ (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Iterator > move\_iterator< _Iterator > operator+ (typename move\_iterator< _Iterator >::difference_type __n, const move\_iterator< _Iterator > &__x)`
- `template<typename _CharT, typename _Traits, typename _Alloc > basic\_string< _CharT, _Traits, _Alloc > operator+ (const basic\_string< _CharT, _Traits, _Alloc > &__lhs, const basic\_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc > basic\_string< _CharT, _Traits, _Alloc > operator+ (const _CharT *__lhs, const basic\_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc > basic\_string< _CharT, _Traits, _Alloc > operator+ (_CharT __lhs, const basic\_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc > basic\_string< _CharT, _Traits, _Alloc > operator+ (const basic\_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc > basic\_string< _CharT, _Traits, _Alloc > operator+ (const basic\_string< _CharT, _Traits, _Alloc > &__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc > basic\_string< _CharT, _Traits, _Alloc > operator+ (basic\_string< _CharT, _Traits, _Alloc > &&__lhs, const basic\_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc > basic\_string< _CharT, _Traits, _Alloc > operator+ (const basic\_string< _CharT, _Traits, _Alloc > &__lhs, basic\_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc > basic\_string< _CharT, _Traits, _Alloc > operator+ (basic\_string< _CharT, _Traits, _Alloc > &&__lhs, basic\_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc > basic\_string< _CharT, _Traits, _Alloc > operator+ (const _CharT *__lhs, basic\_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc > basic\_string< _CharT, _Traits, _Alloc > operator+ (_CharT __lhs, basic\_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc > basic\_string< _CharT, _Traits, _Alloc > operator+ (basic\_string< _CharT, _Traits, _Alloc > &&__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc > basic\_string< _CharT, _Traits, _Alloc > operator+ (basic\_string< _CharT, _Traits, _Alloc > &&__lhs, _CharT __rhs)`
- `ptrdiff_t operator- (const _Bit_iterator_base &__x, const _Bit_iterator_base &__y)`
- `template<typename _Iterator > reverse\_iterator< _Iterator >::difference_type operator- (const reverse\_iterator< _Iterator > &__x, const reverse\_iterator< _Iterator > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr > Deque\_iterator< _Tp, _Ref, _Ptr >::difference_type operator- (const Deque\_iterator< _Tp, _Ref, _Ptr > &__x, const Deque\_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR > Deque\_iterator< _Tp, _RefL, _PtrL >::difference_type operator- (const Deque\_iterator< _Tp, _RefL, _PtrL > &__x, const Deque\_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _IteratorL, typename _IteratorR > auto operator- (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y) -> decltype(__y.base()-__x.base())`

- `template<class _Dom > _Expr< _BinClos< __minus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __minus, typename _Dom::value_type >::result_type > operator- (const typename _Dom::value_type & __t, const _Expr< _Dom, typename _Dom::value_type > & __v)`
- `template<class _Dom > _Expr< _BinClos< __minus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __minus, typename _Dom::value_type >::result_type > operator- (const valarray< typename _Dom::value_type > & __v, const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<class _Dom1, class _Dom2 > _Expr< _BinClos< __minus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __minus, typename _Dom1::value_type >::result_type > operator- (const _Expr< _Dom1, typename _Dom1::value_type > & __v, const _Expr< _Dom2, typename _Dom2::value_type > & __w)`
- `template<class _Dom > _Expr< _BinClos< __minus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __minus, typename _Dom::value_type >::result_type > operator- (const _Expr< _Dom, typename _Dom::value_type > & __v, const typename _Dom::value_type & __t)`
- `template<class _Dom > _Expr< _BinClos< __minus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __minus, typename _Dom::value_type >::result_type > operator- (const _Expr< _Dom, typename _Dom::value_type > & __e, const valarray< typename _Dom::value_type > & __v)`
- `template<typename _IteratorL, typename _IteratorR > auto operator- (const move\_iterator< _IteratorL > & __x, const move\_iterator< _IteratorR > & __y) -> decltype(__x.base()-__y.base())`
- `template<typename _Iterator > auto operator- (const move\_iterator< _Iterator > & __x, const move\_iterator< _Iterator > & __y) -> decltype(__x.base()-__y.base())`
- `template<class _Dom > _Expr< _BinClos< __divides, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __divides, typename _Dom::value_type >::result_type > operator/ (const valarray< typename _Dom::value_type > & __v, const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<class _Dom1, class _Dom2 > _Expr< _BinClos< __divides, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __divides, typename _Dom1::value_type >::result_type > operator/ (const _Expr< _Dom1, typename _Dom1::value_type > & __v, const _Expr< _Dom2, typename _Dom2::value_type > & __w)`
- `template<class _Dom > _Expr< _BinClos< __divides, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __divides, typename _Dom::value_type >::result_type > operator/ (const _Expr< _Dom, typename _Dom::value_type > & __v, const typename _Dom::value_type & __t)`
- `template<class _Dom > _Expr< _BinClos< __divides, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __divides, typename _Dom::value_type >::result_type > operator/ (const typename _Dom::value_type & __t, const _Expr< _Dom, typename _Dom::value_type > & __v)`
- `template<class _Dom > _Expr< _BinClos< __divides, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __divides, typename _Dom::value_type >::result_type > operator/ (const _Expr< _Dom, typename _Dom::value_type > & __e, const valarray< typename _Dom::value_type > & __v)`
- `template<class _T1, class _T2 > constexpr bool operator< (const pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y)`
- `template<typename _Tp, typename _Seq > bool operator< (const stack< _Tp, _Seq > & __x, const stack< _Tp, _Seq > & __y)`
- `template<typename _Tp, typename _Ref, typename _Ptr > bool operator< (const Deque\_iterator< _Tp, _Ref, _Ptr > & __x, const Deque\_iterator< _Tp, _Ref, _Ptr > & __y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR > bool operator< (const Deque\_iterator< _Tp, _RefL, _PtrL > & __x, const Deque\_iterator< _Tp, _RefR, _PtrR > & __y)`
- `template<typename _Tp, typename _Seq > bool operator< (const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq > & __y)`
- `template<typename _Iterator > bool operator< (const reverse\_iterator< _Iterator > & __x, const reverse\_iterator< _Iterator > & __y)`
- `template<typename _IteratorL, typename _IteratorR > bool operator< (const reverse\_iterator< _IteratorL > & __x, const reverse\_iterator< _IteratorR > & __y)`
- `template<typename _Tp1, typename _Tp2 > bool operator< (const shared\_ptr< _Tp1 > & __a, const shared\_ptr< _Tp2 > & __b) noexcept`
- `template<typename _Tp > bool operator< (const shared\_ptr< _Tp > & __a, nullptr_t) noexcept`
- `template<typename _Tp > bool operator< (nullptr_t, const shared\_ptr< _Tp > & __a) noexcept`

- `template<class _Dom > _Expr< _BinClos< __less, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __less, typename _Dom::value_type >::result_type > operator< (const typename _Dom::value_type & __t, const _Expr< _Dom, typename _Dom::value_type > & __v)`
- `template<class _Dom > _Expr< _BinClos< __less, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __less, typename _Dom::value_type >::result_type > operator< (const valarray< typename _Dom::value_type > & __v, const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<class _Dom > _Expr< _BinClos< __less, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __less, typename _Dom::value_type >::result_type > operator< (const _Expr< _Dom, typename _Dom::value_type > & __e, const valarray< typename _Dom::value_type > & __v)`
- `template<class _Dom > _Expr< _BinClos< __less, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __less, typename _Dom::value_type >::result_type > operator< (const _Expr< _Dom, typename _Dom::value_type > & __v, const typename _Dom::value_type & __t)`
- `template<class _Dom1, class _Dom2 > _Expr< _BinClos< __less, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __less, typename _Dom1::value_type >::result_type > operator< (const _Expr< _Dom1, typename _Dom1::value_type > & __v, const _Expr< _Dom2, typename _Dom2::value_type > & __w)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep > bool operator< (const unique_ptr< _Tp, _Dp > & __x, const unique_ptr< _Up, _Ep > & __y)`
- `template<typename _Tp, typename _Dp > bool operator< (const unique_ptr< _Tp, _Dp > & __x, nullptr_t)`
- `template<typename _Tp, typename _Dp > bool operator< (nullptr_t, const unique_ptr< _Tp, _Dp > & __x)`
- `template<typename _Key, typename _Compare, typename _Alloc > bool operator< (const multiset< _Key, _Compare, _Alloc > & __x, const multiset< _Key, _Compare, _Alloc > & __y)`
- `template<typename _Key, typename _Compare, typename _Alloc > bool operator< (const set< _Key, _Compare, _Alloc > & __x, const set< _Key, _Compare, _Alloc > & __y)`
- `template<typename _Bilter > bool operator< (const sub_match< _Bilter > & __lhs, const sub_match< _Bilter > & __rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc > bool operator< (const multimap< _Key, _Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp, _Compare, _Alloc > & __y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc > bool operator< (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > & __x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > & __y)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc > bool operator< (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __lhs, const sub_match< _Bi_iter > & __rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc > bool operator< (const map< _Key, _Tp, _Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare, _Alloc > & __y)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc > bool operator< (const sub_match< _Bi_iter > & __lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __rhs)`
- `template<typename _IteratorL, typename _IteratorR > bool operator< (const move_iterator< _IteratorL > & __x, const move_iterator< _IteratorR > & __y)`
- `template<typename _Iterator > bool operator< (const move_iterator< _Iterator > & __x, const move_iterator< _Iterator > & __y)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp> bool operator< (const __shared_ptr< _Tp1, _Lp > & __a, const __shared_ptr< _Tp2, _Lp > & __b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp> bool operator< (const __shared_ptr< _Tp, _Lp > & __a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp> bool operator< (nullptr_t, const __shared_ptr< _Tp, _Lp > & __a) noexcept`
- `template<typename _Bi_iter > bool operator< (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter > & __rhs)`
- `template<typename _Bi_iter > bool operator< (const sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter >::value_type const * __rhs)`
- `template<typename _Bi_iter > bool operator< (typename iterator_traits< _Bi_iter >::value_type const & __lhs, const sub_match< _Bi_iter > & __rhs)`

- `template<typename _Bi_iter> bool operator< (const sub_match< _Bi_iter> &__lhs, typename iterator_traits< _Bi_iter>::value_type const &__rhs)`
- `template<typename _Tp, typename _Alloc> bool operator< (const forward_list< _Tp, _Alloc> &__lx, const forward_list< _Tp, _Alloc> &__ly)`
- `template<typename _Tp, typename _Alloc> bool operator< (const vector< _Tp, _Alloc> &__x, const vector< _Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc> bool operator< (const list< _Tp, _Alloc> &__x, const list< _Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc> bool operator< (const deque< _Tp, _Alloc> &__x, const deque< _Tp, _Alloc> &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc> bool operator< (const basic_string< _CharT, _Traits, _Alloc> &__lhs, const basic_string< _CharT, _Traits, _Alloc> &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc> bool operator< (const basic_string< _CharT, _Traits, _Alloc> &__lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc> bool operator< (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc> &__rhs)`
- `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp> std::basic_ostream< _Ch, _Tr> & operator<< (std::basic_ostream< _Ch, _Tr> &__os, const __shared_ptr< _Tp, _Lp> &__p)`
- `template<class _Dom> _Expr< _BinClos< __shift_left, _Expr, _ValArray, _Dom, typename _Dom::value_type>, typename __fun< __shift_left, typename _Dom::value_type>::result_type> operator<< (const _Expr< __shift_left, _Dom, typename _Dom::value_type> &__e, const valarray< typename _Dom::value_type> &__v)`
- `template<class _Dom1, class _Dom2> _Expr< _BinClos< __shift_left, _Expr, _Expr, _Dom1, _Dom2>, typename __fun< __shift_left, typename _Dom1::value_type>::result_type> operator<< (const _Expr< _Dom1, typename _Dom1::value_type> &__v, const _Expr< _Dom2, typename _Dom2::value_type> &__w)`
- `template<class _Dom> _Expr< _BinClos< __shift_left, _Expr, _Constant, _Dom, typename _Dom::value_type>, typename __fun< __shift_left, typename _Dom::value_type>::result_type> operator<< (const _Expr< _Dom, typename _Dom::value_type> &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom> _Expr< _BinClos< __shift_left, _Constant, _Expr, typename _Dom::value_type, _Dom>, typename __fun< __shift_left, typename _Dom::value_type>::result_type> operator<< (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type> &__v)`
- `template<class _Dom> _Expr< _BinClos< __shift_left, _ValArray, _Expr, typename _Dom::value_type, _Dom>, typename __fun< __shift_left, typename _Dom::value_type>::result_type> operator<< (const valarray< typename _Dom::value_type> &__v, const _Expr< _Dom, typename _Dom::value_type> &__e)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits> & operator<< (std::basic_ostream< _CharT, _Traits> &__os, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType> &__x)`
- `template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter> basic_ostream< _Ch_type, _Ch_traits> & operator<< (basic_ostream< _Ch_type, _Ch_traits> &__os, const sub_match< _Bi_iter> &__m)`
- `template<typename _IntType, typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits> & operator<< (std::basic_ostream< _CharT, _Traits> &__, const std::uniform_int_distribution< _IntType> &__)`
- `template<typename _RealType, typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits> & operator<< (std::basic_ostream< _CharT, _Traits> &__, const std::uniform_real_distribution< _RealType> &__)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base> basic_ostream< _CharT, _Traits> & operator<< (basic_ostream< _CharT, _Traits> &__os, const __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base> &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc> basic_ostream< _CharT, _Traits> & operator<< (basic_ostream< _CharT, _Traits> &__os, const basic_string< _CharT, _Traits, _Alloc> &__str)`
- `template<typename _RealType, typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits> & operator<< (std::basic_ostream< _CharT, _Traits> &__os, const std::cauchy_distribution< _RealType> &__x)`
- `template<typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits> & operator<< (std::basic_ostream< _CharT, _Traits> &__os, const std::bernoulli_distribution &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits> & operator<< (std::basic_ostream< _CharT, _Traits> &__os, const std::geometric_distribution< _IntType> &__x)`

- `template<typename _RealType, typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits> & operator<<< (std::basic_ostream< _CharT, _Traits> &__os, const std::exponential_distribution< _RealType> &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits> & operator<<< (std::basic_ostream< _CharT, _Traits> &__os, const std::weibull_distribution< _RealType> &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits> & operator<<< (std::basic_ostream< _CharT, _Traits> &__os, const std::extreme_value_distribution< _RealType> &__x)`
- `template<class _T1, class _T2> constexpr bool operator<= (const pair< _T1, _T2> &__x, const pair< _T1, _T2> &__y)`
- `template<typename _Tp, typename _Seq> bool operator<= (const stack< _Tp, _Seq> &__x, const stack< _Tp, _Seq> &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr> bool operator<= (const Deque_iterator< _Tp, _Ref, _Ptr> &__x, const Deque_iterator< _Tp, _Ref, _Ptr> &__y)`
- `template<typename _Tp, typename _Seq> bool operator<= (const queue< _Tp, _Seq> &__x, const queue< _Tp, _Seq> &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR> bool operator<= (const Deque_iterator< _Tp, _RefL, _PtrL> &__x, const Deque_iterator< _Tp, _RefR, _PtrR> &__y)`
- `template<typename _Iterator> bool operator<= (const reverse_iterator< _Iterator> &__x, const reverse_iterator< _Iterator> &__y)`
- `template<typename _IteratorL, typename _IteratorR> bool operator<= (const reverse_iterator< _IteratorL> &__x, const reverse_iterator< _IteratorR> &__y)`
- `template<typename _Tp1, typename _Tp2> bool operator<= (const shared_ptr< _Tp1> &__a, const shared_ptr< _Tp2> &__b) noexcept`
- `template<typename _Tp> bool operator<= (const shared_ptr< _Tp> &__a, nullptr_t) noexcept`
- `template<typename _Tp> bool operator<= (nullptr_t, const shared_ptr< _Tp> &__a) noexcept`
- `template<class _Dom1, class _Dom2> _Expr< _BinClos< __less_equal, _Expr, _Expr, _Dom1, _Dom2>, typename __fun< __less_equal, typename _Dom1::value_type>::result_type> operator<= (const _Expr< _Dom1, typename _Dom1::value_type> &__v, const _Expr< _Dom2, typename _Dom2::value_type> &__w)`
- `template<class _Dom> _Expr< _BinClos< __less_equal, _Constant, _Expr, typename _Dom::value_type, _Dom>, typename __fun< __less_equal, typename _Dom::value_type>::result_type> operator<= (const typename __fun< __less_equal, typename _Dom::value_type>::result_type &__t, const _Expr< _Dom, typename _Dom::value_type> &__v)`
- `template<class _Dom> _Expr< _BinClos< __less_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom>, typename __fun< __less_equal, typename _Dom::value_type>::result_type> operator<= (const valarray< typename _Dom::value_type> &__v, const _Expr< _Dom, typename _Dom::value_type> &__e)`
- `template<class _Dom> _Expr< _BinClos< __less_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type>, typename __fun< __less_equal, typename _Dom::value_type>::result_type> operator<= (const _Expr< _Dom, typename _Dom::value_type> &__e, const valarray< typename _Dom::value_type> &__v)`
- `template<class _Dom> _Expr< _BinClos< __less_equal, _Expr, _Constant, _Dom, typename _Dom::value_type>, typename __fun< __less_equal, typename _Dom::value_type>::result_type> operator<= (const _Expr< _Dom, typename _Dom::value_type> &__v, const typename _Dom::value_type &__t)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep> bool operator<= (const unique_ptr< _Tp, _Dp> &__x, const unique_ptr< _Up, _Ep> &__y)`
- `template<typename _Tp, typename _Dp> bool operator<= (const unique_ptr< _Tp, _Dp> &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp> bool operator<= (nullptr_t, const unique_ptr< _Tp, _Dp> &__x)`
- `template<typename _Key, typename _Compare, typename _Alloc> bool operator<= (const multiset< _Key, _Compare, _Alloc> &__x, const multiset< _Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc> bool operator<= (const set< _Key, _Compare, _Alloc> &__x, const set< _Key, _Compare, _Alloc> &__y)`
- `template<typename _Bilter> bool operator<= (const sub_match< _Bilter> &__lhs, const sub_match< _Bilter> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool operator<= (const multimap< _Key, _Tp, _Compare, _Alloc> &__x, const multimap< _Key, _Tp, _Compare, _Alloc> &__y)`



- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc> bool operator<= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc> &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc> &__y)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> bool operator<= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc> &__lhs, const sub_match< _Bi_iter> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool operator<= (const map< _Key, _Tp, _Compare, _Alloc> &__x, const map< _Key, _Tp, _Compare, _Alloc> &__y)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc> bool operator<= (const sub_match< _Bi_iter> &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc> &__rhs)`
- `template<typename _IteratorL, typename _IteratorR> bool operator<= (const move_iterator< _IteratorL> &__x, const move_iterator< _IteratorR> &__y)`
- `template<typename _Iterator> bool operator<= (const move_iterator< _Iterator> &__x, const move_iterator< _Iterator> &__y)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp> bool operator<= (const __shared_ptr< _Tp1, _Lp> &__a, const __shared_ptr< _Tp2, _Lp> &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp> bool operator<= (const __shared_ptr< _Tp, _Lp> &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp> bool operator<= (nullptr_t, const __shared_ptr< _Tp, _Lp> &__a) noexcept`
- `template<typename _Bi_iter> bool operator<= (typename iterator_traits< _Bi_iter>::value_type const *__lhs, const sub_match< _Bi_iter> &__rhs)`
- `template<typename _Bi_iter> bool operator<= (const sub_match< _Bi_iter> &__lhs, typename iterator_traits< _Bi_iter>::value_type const *__rhs)`
- `template<typename _Bi_iter> bool operator<= (typename iterator_traits< _Bi_iter>::value_type const &__lhs, const sub_match< _Bi_iter> &__rhs)`
- `template<typename _Bi_iter> bool operator<= (const sub_match< _Bi_iter> &__lhs, typename iterator_traits< _Bi_iter>::value_type const &__rhs)`
- `template<typename _Tp, typename _Alloc> bool operator<= (const forward_list< _Tp, _Alloc> &__lx, const forward_list< _Tp, _Alloc> &__ly)`
- `template<typename _Tp, typename _Alloc> bool operator<= (const vector< _Tp, _Alloc> &__x, const vector< _Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc> bool operator<= (const list< _Tp, _Alloc> &__x, const list< _Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc> bool operator<= (const deque< _Tp, _Alloc> &__x, const deque< _Tp, _Alloc> &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc> bool operator<= (const basic_string< _CharT, _Traits, _Alloc> &__lhs, const basic_string< _CharT, _Traits, _Alloc> &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc> bool operator<= (const basic_string< _CharT, _Traits, _Alloc> &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc> bool operator<= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc> &__rhs)`
- `template<typename _T1, typename _T2> bool operator== (const allocator< _T1> &, const allocator< _T2> &)`
- `template<typename _Tp, typename _CharT, typename _Traits, typename _Dist> bool operator== (const istream_iterator< _Tp, _CharT, _Traits, _Dist> &__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist> &__y)`
- `template<typename _Tp> bool operator== (const allocator< _Tp> &, const allocator< _Tp> &)`
- `template<typename _CharT, typename _Traits> bool operator== (const istreambuf_iterator< _CharT, _Traits> &__a, const istreambuf_iterator< _CharT, _Traits> &__b)`
- `template<class _T1, class _T2> constexpr bool operator== (const pair< _T1, _T2> &__x, const pair< _T1, _T2> &__y)`
- `template<typename _StateT> bool operator== (const fpos< _StateT> &__lhs, const fpos< _StateT> &__rhs)`
- `template<typename _Tp, typename _Seq> bool operator== (const stack< _Tp, _Seq> &__x, const stack< _Tp, _Seq> &__y)`

- `template<typename _Tp, typename _Ref, typename _Ptr> bool operator== (const Deque\_iterator< _Tp, _Ref, _Ptr > &__x, const Deque\_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR> bool operator== (const Deque\_iterator< _Tp, _RefL, _PtrL > &__x, const Deque\_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp> bool operator== (const Fwd\_list\_iterator< _Tp > &__x, const Fwd\_list\_const\_iterator< _Tp > &__y)`
- `template<typename _Tp, typename _Seq> bool operator== (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Val> bool operator== (const List\_iterator< _Val > &__x, const List\_const\_iterator< _Val > &__y)`
- `template<typename _Iterator> bool operator== (const reverse\_iterator< _Iterator > &__x, const reverse\_iterator< _Iterator > &__y)`
- `template<typename _Val> bool operator== (const Rb\_tree\_iterator< _Val > &__x, const Rb\_tree\_const\_iterator< _Val > &__y)`
- `template<typename _Tp1, typename _Tp2> bool operator== (const shared\_ptr< _Tp1 > &__a, const shared\_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp> bool operator== (const shared\_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp> bool operator== (nullptr_t, const shared\_ptr< _Tp > &__a) noexcept`
- `template<typename _IteratorL, typename _IteratorR> bool operator== (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)`
- `template<class _Dom> _Expr< _BinClos< __equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __equal_to, typename _Dom::value_type>::result_type> operator== (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom> _Expr< _BinClos< __equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __equal_to, typename _Dom::value_type>::result_type> operator== (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2> _Expr< _BinClos< __equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __equal_to, typename _Dom1::value_type>::result_type> operator== (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom> _Expr< _BinClos< __equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __equal_to, typename _Dom::value_type>::result_type> operator== (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom> _Expr< _BinClos< __equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __equal_to, typename _Dom::value_type>::result_type> operator== (const typename __fun< __equal_to, typename _Dom::value_type>::result_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep> bool operator== (const unique\_ptr< _Tp, _Dp > &__x, const unique\_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp> bool operator== (const unique\_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<typename _Tp, typename _Dp> bool operator== (nullptr_t, const unique\_ptr< _Tp, _Dp > &__x) noexcept`
- `template<typename _Key, typename _Compare, typename _Alloc> bool operator== (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc> bool operator== (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Bilter> bool operator== (const sub\_match< _Bilter > &__lhs, const sub\_match< _Bilter > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool operator== (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc> bool operator== (const Rb\_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const Rb\_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> bool operator== (const sub\_match\_string< __fun< __equal_to, typename _Ch_traits::value_type>::result_type> &__lhs, const sub\_match< _Bi_iter > &__rhs)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool operator== (const map< _Key, _Tp, _Compare, _Alloc> &__x, const map< _Key, _Tp, _Compare, _Alloc> &__y)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> bool operator== (const sub_match< _Bi_iter> &__lhs, const sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc> &__rhs)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp> bool operator== (const __shared_ptr< _Tp1, _Lp> &__a, const __shared_ptr< _Tp2, _Lp> &__b) noexcept`
- `template<typename _IteratorL, typename _IteratorR> bool operator== (const move_iterator< _IteratorL> &__x, const move_iterator< _IteratorR> &__y)`
- `template<typename _Tp, _Lock_policy _Lp> bool operator== (const __shared_ptr< _Tp, _Lp> &__a, nullptr_t) noexcept`
- `template<typename _Iterator> bool operator== (const move_iterator< _Iterator> &__x, const move_iterator< _Iterator> &__y)`
- `template<typename _Tp, _Lock_policy _Lp> bool operator== (nullptr_t, const __shared_ptr< _Tp, _Lp> &__a) noexcept`
- `template<typename _Bi_iter> bool operator== (typename iterator_traits< _Bi_iter>::value_type const * __lhs, const sub_match< _Bi_iter> &__rhs)`
- `template<typename _Bi_iter> bool operator== (const sub_match< _Bi_iter> &__lhs, typename iterator_traits< _Bi_iter>::value_type const * __rhs)`
- `template<typename _Bi_iter> bool operator== (typename iterator_traits< _Bi_iter>::value_type const &__lhs, const sub_match< _Bi_iter> &__rhs)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc> bool operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc> &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc> &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc> bool operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc> &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Bi_iter> bool operator== (const sub_match< _Bi_iter> &__lhs, typename iterator_traits< _Bi_iter>::value_type const &__rhs)`
- `template<typename _Tp, typename _Alloc> bool operator== (const forward_list< _Tp, _Alloc> &__lx, const forward_list< _Tp, _Alloc> &__ly)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc> bool operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc> &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc> bool operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc> &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc> bool operator== (const vector< _Tp, _Alloc> &__x, const vector< _Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc> bool operator== (const list< _Tp, _Alloc> &__x, const list< _Tp, _Alloc> &__y)`
- `template<typename _Bi_iter, typename _Alloc> bool operator== (const match_results< _Bi_iter, _Alloc> &__m1, const match_results< _Bi_iter, _Alloc> &__m2)`
- `template<typename _Tp, typename _Alloc> bool operator== (const deque< _Tp, _Alloc> &__x, const deque< _Tp, _Alloc> &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc> bool operator== (const basic_string< _CharT, _Traits, _Alloc> &__lhs, const basic_string< _CharT, _Traits, _Alloc> &__rhs)`
- `template<typename _CharT> __gnu_cxx::enable_if< __is_char< _CharT>::__value, bool>::__type operator== (const basic_string< _CharT> &__lhs, const basic_string< _CharT> &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc> bool operator== (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc> &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc> bool operator== (const basic_string< _CharT, _Traits, _Alloc> &__lhs, const _CharT * __rhs)`
- `template<class _T1, class _T2> constexpr bool operator> (const pair< _T1, _T2> &__x, const pair< _T1, _T2> &__y)`
- `template<typename _Tp, typename _Seq> bool operator> (const stack< _Tp, _Seq> &__x, const stack< _Tp, _Seq> &__y)`



- `template<typename _Tp, typename _Ref, typename _Ptr> bool operator> (const Deque\_iterator< _Tp, _Ref, _Ptr > &__x, const Deque\_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR> bool operator> (const Deque\_iterator< _Tp, _RefL, _PtrL > &__x, const Deque\_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Seq> bool operator> (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Iterator> bool operator> (const reverse\_iterator< _Iterator > &__x, const reverse\_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR> bool operator> (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)`
- `template<typename _Tp1, typename _Tp2> bool operator> (const shared\_ptr< _Tp1 > &__a, const shared\_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp> bool operator> (const shared\_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp> bool operator> (nullptr_t, const shared\_ptr< _Tp > &__a) noexcept`
- `template<class _Dom> _Expr< _BinClos< __greater, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __greater, typename _Dom::value_type >::result_type > operator> (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom> _Expr< _BinClos< __greater, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __greater, typename _Dom::value_type >::result_type > operator> (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2> _Expr< _BinClos< __greater, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __greater, typename _Dom1::value_type >::result_type > operator> (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom> _Expr< _BinClos< __greater, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __greater, typename _Dom::value_type >::result_type > operator> (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom> _Expr< _BinClos< __greater, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __greater, typename _Dom::value_type >::result_type > operator> (const typename __fun< __greater, typename _Dom::value_type >::result_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep> bool operator> (const unique\_ptr< _Tp, _Dp > &__x, const unique\_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp> bool operator> (const unique\_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp> bool operator> (nullptr_t, const unique\_ptr< _Tp, _Dp > &__x)`
- `template<typename _Key, typename _Compare, typename _Alloc> bool operator> (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc> bool operator> (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool operator> (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Bilter> bool operator> (const sub\_match< _Bilter > &__lhs, const sub\_match< _Bilter > &__rhs)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc> bool operator> (const Rb\_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const Rb\_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> bool operator> (const sub\_match\_string< __fun< __greater, typename _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub\_match< _Bi_iter > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool operator> (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc> bool operator> (const sub\_match< _Bi_iter > &__lhs, const sub\_match\_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR> bool operator> (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)`

- `template<typename _Iterator> bool operator> (const move_iterator< _Iterator> &__x, const move_iterator< _Iterator> &__y)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp> bool operator> (const __shared_ptr< _Tp1, _Lp> &__a, const __shared_ptr< _Tp2, _Lp> &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp> bool operator> (const __shared_ptr< _Tp, _Lp> &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp> bool operator> (nullptr_t, const __shared_ptr< _Tp, _Lp> &__a) noexcept`
- `template<typename _Bi_iter> bool operator> (typename iterator_traits< _Bi_iter>::value_type const *__lhs, const sub_match< _Bi_iter> &__rhs)`
- `template<typename _Bi_iter> bool operator> (const sub_match< _Bi_iter> &__lhs, typename iterator_traits< _Bi_iter>::value_type const *__rhs)`
- `template<typename _Bi_iter> bool operator> (typename iterator_traits< _Bi_iter>::value_type const &__lhs, const sub_match< _Bi_iter> &__rhs)`
- `template<typename _Bi_iter> bool operator> (const sub_match< _Bi_iter> &__lhs, typename iterator_traits< _Bi_iter>::value_type const &__rhs)`
- `template<typename _Tp, typename _Alloc> bool operator> (const forward_list< _Tp, _Alloc> &__lx, const forward_list< _Tp, _Alloc> &__ly)`
- `template<typename _Tp, typename _Alloc> bool operator> (const vector< _Tp, _Alloc> &__x, const vector< _Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc> bool operator> (const list< _Tp, _Alloc> &__x, const list< _Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc> bool operator> (const deque< _Tp, _Alloc> &__x, const deque< _Tp, _Alloc> &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc> bool operator> (const basic_string< _CharT, _Traits, _Alloc> &__lhs, const basic_string< _CharT, _Traits, _Alloc> &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc> bool operator> (const basic_string< _CharT, _Traits, _Alloc> &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc> bool operator> (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc> &__rhs)`
- `template<class _T1, class _T2> constexpr bool operator>= (const pair< _T1, _T2> &__x, const pair< _T1, _T2> &__y)`
- `template<typename _Tp, typename _Seq> bool operator>= (const stack< _Tp, _Seq> &__x, const stack< _Tp, _Seq> &__y)`
- `template<typename _Tp, typename _Seq> bool operator>= (const queue< _Tp, _Seq> &__x, const queue< _Tp, _Seq> &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr> bool operator>= (const _Deque_iterator< _Tp, _Ref, _Ptr> &__x, const _Deque_iterator< _Tp, _Ref, _Ptr> &__y)`
- `template<typename _Iterator> bool operator>= (const reverse_iterator< _Iterator> &__x, const reverse_iterator< _Iterator> &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR> bool operator>= (const _Deque_iterator< _Tp, _RefL, _PtrL> &__x, const _Deque_iterator< _Tp, _RefR, _PtrR> &__y)`
- `template<typename _IteratorL, typename _IteratorR> bool operator>= (const reverse_iterator< _IteratorL> &__x, const reverse_iterator< _IteratorR> &__y)`
- `template<typename _Tp1, typename _Tp2> bool operator>= (const shared_ptr< _Tp1> &__a, const shared_ptr< _Tp2> &__b) noexcept`
- `template<typename _Tp> bool operator>= (const shared_ptr< _Tp> &__a, nullptr_t) noexcept`
- `template<class _Dom> _Expr< _BinClos< __greater_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type>, typename __fun< __greater_equal, typename _Dom::value_type>::result_type> operator>= (const _Expr< _Dom, typename _Dom::value_type> &__e, const valarray< typename _Dom::value_type> &__v)`
- `template<class _Dom> _Expr< _BinClos< __greater_equal, _Constant, _Expr, typename _Dom::value_type, _Dom>, typename __fun< __greater_equal, typename _Dom::value_type>::result_type> operator>= (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type> &__v)`

- `template<class _Dom1, class _Dom2 > _Expr< _BinClos< __greater_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __greater_equal, typename _Dom1::value_type >::result_type > operator>= (const _Expr< _BinClos< __greater_equal, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom > _Expr< _BinClos< __greater_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __greater_equal, typename _Dom::value_type >::result_type > operator>= (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom > _Expr< _BinClos< __greater_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __greater_equal, typename _Dom::value_type >::result_type > operator>= (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<typename _Tp > bool operator>= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep > bool operator>= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp > bool operator>= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp > bool operator>= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Key, typename _Compare, typename _Alloc > bool operator>= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc > bool operator>= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Bilter > bool operator>= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc > bool operator>= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc > bool operator>= (const Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc > bool operator>= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc > bool operator>= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc > bool operator>= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR > bool operator>= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator > bool operator>= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp> bool operator>= (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp> bool operator>= (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp> bool operator>= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Bi_iter > bool operator>= (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter > bool operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const * __rhs)`
- `template<typename _Bi_iter > bool operator>= (typename iterator_traits< _Bi_iter >::value_type const & __lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter > bool operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const & __rhs)`
- `template<typename _Tp, typename _Alloc > bool operator>= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc > bool operator>= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`

- `template<typename _Tp, typename _Alloc> bool operator>= (const list< _Tp, _Alloc> &__x, const list< _Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc> bool operator>= (const deque< _Tp, _Alloc> &__x, const deque< _Tp, _Alloc> &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc> bool operator>= (const basic_string< _CharT, _Traits, _Alloc> &__lhs, const basic_string< _CharT, _Traits, _Alloc> &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc> bool operator>= (const basic_string< _CharT, _Traits, _Alloc> &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc> bool operator>= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc> &__rhs)`
- `template<class _Dom> _Expr< _BinClos< __shift_right, _Expr, _Constant, _Dom, typename _Dom::value_type>, typename __fun< __shift_right, typename _Dom::value_type>::result_type> operator>> (const _Expr< _Dom, typename _Dom::value_type> &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom> _Expr< _BinClos< __shift_right, _Constant, _Expr, typename _Dom::value_type, _Dom>, typename __fun< __shift_right, typename _Dom::value_type>::result_type> operator>> (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type> &__v)`
- `template<class _Dom1, class _Dom2> _Expr< _BinClos< __shift_right, _Expr, _Expr, _Dom1, _Dom2>, typename __fun< __shift_right, typename _Dom1::value_type>::result_type> operator>> (const _Expr< _Dom1, typename _Dom1::value_type> &__v, const _Expr< _Dom2, typename _Dom2::value_type> &__w)`
- `template<class _Dom> _Expr< _BinClos< __shift_right, _ValArray, _Expr, typename _Dom::value_type, _Dom>, typename __fun< __shift_right, typename _Dom::value_type>::result_type> operator>> (const valarray< typename _Dom::value_type> &__v, const _Expr< _Dom, typename _Dom::value_type> &__e)`
- `template<class _Dom> _Expr< _BinClos< __shift_right, _Expr, _ValArray, _Dom, typename _Dom::value_type>, typename __fun< __shift_right, typename _Dom::value_type>::result_type> operator>> (const _Expr< __shift_right, _Dom, typename _Dom::value_type> &__e, const valarray< typename _Dom::value_type> &__v)`
- `template<typename _IntType, typename _CharT, typename _Traits> std::basic_istream< _CharT, _Traits> & operator>> (std::basic_istream< _CharT, _Traits> &, std::uniform_int_distribution< _IntType> &)`
- `template<typename _RealType, typename _CharT, typename _Traits> std::basic_istream< _CharT, _Traits> & operator>> (std::basic_istream< _CharT, _Traits> &, std::uniform_real_distribution< _RealType> &)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename> class _Base> basic_istream< _CharT, _Traits> & operator>> (basic_istream< _CharT, _Traits> &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base> &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc> basic_istream< _CharT, _Traits> & operator>> (basic_istream< _CharT, _Traits> &__is, basic_string< _CharT, _Traits, _Alloc> &__str)`
- `template<> basic_istream< char> & operator>> (basic_istream< char> &__is, basic_string< char> &__str)`
- `template<typename _RealType, typename _CharT, typename _Traits> std::basic_istream< _CharT, _Traits> & operator>> (std::basic_istream< _CharT, _Traits> &__is, std::cauchy_distribution< _RealType> &__x)`
- `template<typename _CharT, typename _Traits> std::basic_istream< _CharT, _Traits> & operator>> (std::basic_istream< _CharT, _Traits> &__is, std::bernoulli_distribution &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits> std::basic_istream< _CharT, _Traits> & operator>> (std::basic_istream< _CharT, _Traits> &__is, std::geometric_distribution< _IntType> &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits> std::basic_istream< _CharT, _Traits> & operator>> (std::basic_istream< _CharT, _Traits> &__is, std::exponential_distribution< _RealType> &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits> std::basic_istream< _CharT, _Traits> & operator>> (std::basic_istream< _CharT, _Traits> &__is, std::weibull_distribution< _RealType> &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits> std::basic_istream< _CharT, _Traits> & operator>> (std::basic_istream< _CharT, _Traits> &__is, std::extreme_value_distribution< _RealType> &__x)`
- `constexpr _ios_Fmtflags operator^ (_ios_Fmtflags __a, _ios_Fmtflags __b)`
- `constexpr _ios_Openmode operator^ (_ios_Openmode __a, _ios_Openmode __b)`
- `constexpr _ios_ostate operator^ (_ios_ostate __a, _ios_ostate __b)`
- `template<class _Dom> _Expr< _BinClos< __bitwise_xor, _Constant, _Expr, typename _Dom::value_type, _Dom>, typename __fun< __bitwise_xor, typename _Dom::value_type>::result_type> operator^ (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type> &__v)`

- `template<class _Dom > _Expr< _BinClos< __bitwise_xor, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __bitwise_xor, typename _Dom::value_type >::result_type > operator^ (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom > _Expr< _BinClos< __bitwise_xor, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __bitwise_xor, typename _Dom::value_type >::result_type > operator^ (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 > _Expr< _BinClos< __bitwise_xor, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __bitwise_xor, typename _Dom1::value_type >::result_type > operator^ (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom > _Expr< _BinClos< __bitwise_xor, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __bitwise_xor, typename _Dom::value_type >::result_type > operator^ (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `const _los_Fmtflags & operator^= (_los_Fmtflags &__a, _los_Fmtflags __b)`
- `const _los_Openmode & operator^= (_los_Openmode &__a, _los_Openmode __b)`
- `const _los_losestate & operator^= (_los_losestate &__a, _los_losestate __b)`
- `constexpr memory\_order operator| (memory\_order __m, __memory_order_modifier __mod)`
- `constexpr _los_Fmtflags operator| (_los_Fmtflags __a, _los_Fmtflags __b)`
- `constexpr _los_Openmode operator| (_los_Openmode __a, _los_Openmode __b)`
- `constexpr _los_losestate operator| (_los_losestate __a, _los_losestate __b)`
- `template<class _Dom > _Expr< _BinClos< __bitwise_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __bitwise_or, typename _Dom::value_type >::result_type > operator| (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom > _Expr< _BinClos< __bitwise_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __bitwise_or, typename _Dom::value_type >::result_type > operator| (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom > _Expr< _BinClos< __bitwise_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __bitwise_or, typename _Dom::value_type >::result_type > operator| (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom > _Expr< _BinClos< __bitwise_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __bitwise_or, typename _Dom::value_type >::result_type > operator| (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2 > _Expr< _BinClos< __bitwise_or, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __bitwise_or, typename _Dom1::value_type >::result_type > operator| (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `const _los_Fmtflags & operator|= (_los_Fmtflags &__a, _los_Fmtflags __b)`
- `const _los_Openmode & operator|= (_los_Openmode &__a, _los_Openmode __b)`
- `const _los_losestate & operator|= (_los_losestate &__a, _los_losestate __b)`
- `template<class _Dom1, class _Dom2 > _Expr< _BinClos< __logical_or, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __logical_or, typename _Dom1::value_type >::result_type > operator|| (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom > _Expr< _BinClos< __logical_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __logical_or, typename _Dom::value_type >::result_type > operator|| (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom > _Expr< _BinClos< __logical_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __logical_or, typename _Dom::value_type >::result_type > operator|| (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom > _Expr< _BinClos< __logical_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __logical_or, typename _Dom::value_type >::result_type > operator|| (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom > _Expr< _BinClos< __logical_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __logical_or, typename _Dom::value_type >::result_type > operator|| (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`



- constexpr `_ios_Fmtflags` **operator~** (`_ios_Fmtflags __a`)
- constexpr `_ios_Openmode` **operator~** (`_ios_Openmode __a`)
- constexpr `_ios_istate` **operator~** (`_ios_istate __a`)
- template<typename `_RAIter`> void **partial\_sort** (`_RAIter`, `_RAIter`, `_RAIter`)
- template<typename `_RAIter`, typename `_Compare`> void **partial\_sort** (`_RAIter`, `_RAIter`, `_RAIter`, `_Compare`)
- template<typename `_RandomAccessIterator`> void **partial\_sort** (`_RandomAccessIterator __first`, `_RandomAccessIterator __middle`, `_RandomAccessIterator __last`)
- template<typename `_RandomAccessIterator`, typename `_Compare`> void **partial\_sort** (`_RandomAccessIterator __first`, `_RandomAccessIterator __middle`, `_RandomAccessIterator __last`, `_Compare __comp`)
- template<typename `_Iter`, typename `_RAIter`> `_RAIter` **partial\_sort\_copy** (`_Iter`, `_Iter`, `_RAIter`, `_RAIter`)
- template<typename `_Iter`, typename `_RAIter`, typename `_Compare`> `_RAIter` **partial\_sort\_copy** (`_Iter`, `_Iter`, `_RAIter`, `_RAIter`, `_Compare`)
- template<typename `_InputIterator`, typename `_RandomAccessIterator`> `_RandomAccessIterator` **partial\_sort\_copy** (`_InputIterator __first`, `_InputIterator __last`, `_RandomAccessIterator __result_first`, `_RandomAccessIterator __result_last`)
- template<typename `_InputIterator`, typename `_RandomAccessIterator`, typename `_Compare`> `_RandomAccessIterator` **partial\_sort\_copy** (`_InputIterator __first`, `_InputIterator __last`, `_RandomAccessIterator __result_first`, `_RandomAccessIterator __result_last`, `_Compare __comp`)
- template<typename `_InputIterator`, typename `_OutputIterator`> `_OutputIterator` **partial\_sum** (`_InputIterator __first`, `_InputIterator __last`, `_OutputIterator __result`)
- template<typename `_InputIterator`, typename `_OutputIterator`, typename `_BinaryOperation`> `_OutputIterator` **partial\_sum** (`_InputIterator __first`, `_InputIterator __last`, `_OutputIterator __result`, `_BinaryOperation __binary_op`)
- template<typename `_BIter`, typename `_Predicate`> `_BIter` **partition** (`_BIter`, `_BIter`, `_Predicate`)
- template<typename `_ForwardIterator`, typename `_Predicate`> `_ForwardIterator` **partition** (`_ForwardIterator __first`, `_ForwardIterator __last`, `_Predicate __pred`)
- template<typename `_Iter`, typename `_Olter1`, typename `_Olter2`, typename `_Predicate`> `pair<_Olter1, _Olter2>` **partition\_copy** (`_Iter`, `_Iter`, `_Olter1`, `_Olter2`, `_Predicate`)
- template<typename `_InputIterator`, typename `_OutputIterator1`, typename `_OutputIterator2`, typename `_Predicate`> `pair<_OutputIterator1, _OutputIterator2>` **partition\_copy** (`_InputIterator __first`, `_InputIterator __last`, `_OutputIterator1 __out_true`, `_OutputIterator2 __out_false`, `_Predicate __pred`)
- template<typename `_Filter`, typename `_Predicate`> `_Filter` **partition\_point** (`_Filter`, `_Filter`, `_Predicate`)
- template<typename `_ForwardIterator`, typename `_Predicate`> `_ForwardIterator` **partition\_point** (`_ForwardIterator __first`, `_ForwardIterator __last`, `_Predicate __pred`)
- template<typename `_RandomAccessIterator`> void **pop\_heap** (`_RandomAccessIterator __first`, `_RandomAccessIterator __last`)
- template<typename `_RandomAccessIterator`, typename `_Compare`> void **pop\_heap** (`_RandomAccessIterator __first`, `_RandomAccessIterator __last`, `_Compare __comp`)
- template<typename `_RAIter`> void **pop\_heap** (`_RAIter`, `_RAIter`)
- template<typename `_RAIter`, typename `_Compare`> void **pop\_heap** (`_RAIter`, `_RAIter`, `_Compare`)
- template<typename `_Tp`> `_Expr<_BinClos<_Pow, _Constant, _ValArray, _Tp, _Tp>, _Tp>` **pow** (const `_Tp &__t`, const `valarray<_Tp> &__v`)
- template<class `_Dom`> `_Expr<_BinClos<_Pow, _Expr, _Constant, _Dom, typename _Dom::value_type>, typename _Dom::value_type>` **pow** (const `_Expr<_Dom, typename _Dom::value_type> &__e`, const `typename _Dom::value_type &__t`)
- template<class `_Dom1`, class `_Dom2`> `_Expr<_BinClos<_Pow, _Expr, _Expr, _Dom1, _Dom2>, typename _Dom1::value_type>` **pow** (const `_Expr<_Dom1, typename _Dom1::value_type> &__e1`, const `_Expr<_Dom2, typename _Dom2::value_type> &__e2`)
- template<class `_Dom`> `_Expr<_BinClos<_Pow, _ValArray, _Expr, typename _Dom::value_type, _Dom>, typename _Dom::value_type>` **pow** (const `valarray<typename _Dom::value_type> &__v`, const `_Expr<_Dom, typename _Dom::value_type> &__e`)
- template<typename `_Tp`> `_Expr<_BinClos<_Pow, _ValArray, _ValArray, _Tp, _Tp>, _Tp>` **pow** (const `valarray<_Tp> &__v`, const `valarray<_Tp> &__w`)

- `template<typename _Tp> _Expr< _BinClos< _Pow, _ValArray, _Constant, _Tp, _Tp>, _Tp> pow (const valarray< _Tp> &__v, const _Tp &__t)`
- `template<class _Dom> _Expr< _BinClos< _Pow, _Expr, _ValArray, _Dom, typename _Dom::value_type>, typename _Dom::value_type> pow (const _Expr< _Dom, typename _Dom::value_type> &__e, const valarray< typename _Dom::value_type> &__v)`
- `template<class _Dom> _Expr< _BinClos< _Pow, _Constant, _Expr, typename _Dom::value_type, _Dom>, typename _Dom::value_type> pow (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type> &__e)`
- `template<typename _BidirectionalIterator> _BidirectionalIterator prev (_BidirectionalIterator __x, typename iterator_traits< _BidirectionalIterator>::difference_type __n=1)`
- `template<typename _Blter> bool prev_permutation (_Blter, _Blter)`
- `template<typename _Blter, typename _Compare> bool prev_permutation (_Blter, _Blter, _Compare)`
- `template<typename _BidirectionalIterator> bool prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare> bool prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _Arg, typename _Result> pointer_to_unary_function< _Arg, _Result> ptr_fun (_Result(*__x)(__Arg))`
- `template<typename _Arg1, typename _Arg2, typename _Result> pointer_to_binary_function< _Arg1, _Arg2, _Result> ptr_fun (_Result(*__x)(__Arg1, __Arg2))`
- `template<typename _RandomAccessIterator> void push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare> void push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RAlter> void push_heap (_RAlter, _RAlter)`
- `template<typename _RAlter, typename _Compare> void push_heap (_RAlter, _RAlter, _Compare)`
- `template<typename _RAlter> void random_shuffle (_RAlter, _RAlter)`
- `template<typename _RAlter, typename _Generator> void random_shuffle (_RAlter, _RAlter, _Generator &&)`
- `template<typename _RandomAccessIterator> void random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator> void random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator && __rand)`
- `template<typename _Filter, typename _Tp> _Filter remove (_Filter, _Filter, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp> _ForwardIterator remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _Iter, typename _OIter, typename _Tp> _OIter remove_copy (_Iter, _Iter, _OIter, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp> _OutputIterator remove_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__value)`
- `template<typename _Iter, typename _OIter, typename _Predicate> _OIter remove_copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate> _OutputIterator remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _Filter, typename _Predicate> _Filter remove_if (_Filter, _Filter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate> _ForwardIterator remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _Filter, typename _Tp> void replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp> void replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _Iter, typename _OIter, typename _Tp> _OIter replace_copy (_Iter, _Iter, _OIter, const _Tp &, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp> _OutputIterator replace_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__old_value, const _Tp &__new_value)`

- `template<typename _Iter, typename _OIter, typename _Predicate, typename _Tp> _OIter replace_copy_if (_Iter, _Iter, _OIter, _Predicate, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp> _OutputIterator replace_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp & __new_value)`
- `template<typename _Filter, typename _Predicate, typename _Tp> void replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Tp> void replace_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp & __new_value)`
- `void rethrow_exception (exception_ptr) __attribute__((__noreturn__))`
- `template<typename _Ex> void rethrow_if_nested (const _Ex & __ex)`
- `void rethrow_if_nested (const nested_exception & __ex)`
- `template<typename _Tp> void return_temporary_buffer (_Tp * __p)`
- `template<typename _BIter> void reverse (_BIter, _BIter)`
- `template<typename _BidirectionalIterator> void reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BIter, typename _OIter> _OIter reverse_copy (_BIter, _BIter, _OIter)`
- `template<typename _BidirectionalIterator, typename _OutputIterator> _OutputIterator reverse_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result)`
- `ios_base & right (ios_base & __base)`
- `template<typename _Filter> void rotate (_Filter, _Filter, _Filter)`
- `template<typename _ForwardIterator> void rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last)`
- `template<typename _Filter, typename _OIter> _OIter rotate_copy (_Filter, _Filter, _Filter, _OIter)`
- `template<typename _ForwardIterator, typename _OutputIterator> _OutputIterator rotate_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, _OutputIterator __result)`
- `ios_base & scientific (ios_base & __base)`
- `template<typename _Filter1, typename _Filter2> _Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate> _Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2> _ForwardIterator1 search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate> _ForwardIterator1 search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _Filter, typename _Size, typename _Tp> _Filter search_n (_Filter, _Filter, _Size, const _Tp &)`
- `template<typename _Filter, typename _Size, typename _Tp, typename _BinaryPredicate> _Filter search_n (_Filter, _Filter, _Size, const _Tp &, _BinaryPredicate)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp> _ForwardIterator search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp & __val)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate> _ForwardIterator search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp & __val, _BinaryPredicate __binary_pred)`
- `template<typename _Iter1, typename _Iter2, typename _OIter> _OIter set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare> _OIter set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator> _OutputIterator set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare> _OutputIterator set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`



- `template<typename _Iter1, typename _Iter2, typename _OIter> _OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare> _OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator> _OutputIterator set_intersection (↵  
_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator  
__result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare> _OutputIterator  
set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2,  
_OutputIterator __result, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter> _OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2,  
_Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare> _OIter set_symmetric_difference (↵  
_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator> _OutputIterator set_symmetric ↵  
difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, ↵  
_OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare> _OutputIterator  
set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Input↵  
Iterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter> _OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare> _OIter set_union (_Iter1, _Iter1, ↵  
_Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator> _OutputIterator set_union (_Input↵  
Iterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator ↵  
__result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare> _OutputIterator  
set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, ↵  
_OutputIterator __result, _Compare __comp)`
- `ios_base & showbase (ios_base & __base)`
- `ios_base & showpoint (ios_base & __base)`
- `ios_base & showpos (ios_base & __base)`
- `template<typename _RAIter, typename _UGenerator> void shuffle (_RAIter, _RAIter, _UGenerator &&)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator> void shuffle (_RandomAccess↵  
Iterator __first, _RandomAccessIterator __last, _UniformRandomNumberGenerator && __g)`
- `template<class _Dom> _Expr< _UnClos< _Sin, _Expr, _Dom>, typename _Dom::value_type> sin (const _Expr<  
_Dom, typename _Dom::value_type> & __e)`
- `template<typename _Tp> _Expr< _UnClos< _Sin, _ValArray, _Tp>, _Tp> sin (const valarray< _Tp> & __v)`
- `template<typename _Tp> _Expr< _UnClos< _Sinh, _ValArray, _Tp>, _Tp> sinh (const valarray< _Tp> & __v)`
- `template<class _Dom> _Expr< _UnClos< _Sinh, _Expr, _Dom>, typename _Dom::value_type> sinh (const  
_Expr< _Dom, typename _Dom::value_type> & __e)`
- `ios_base & skipws (ios_base & __base)`
- `template<typename _RAIter> void sort (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare> void sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator> void sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare> void sort (_RandomAccessIterator __first, _Random↵  
AccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator> void sort_heap (_RandomAccessIterator __first, _RandomAccessIterator  
__last)`
- `template<typename _RandomAccessIterator, typename _Compare> void sort_heap (_RandomAccessIterator __first, ↵  
_RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RAIter> void sort_heap (_RAIter, _RAIter)`

- `template<typename _RAIter, typename _Compare> void sort_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _Tp> _Expr<_UnClos<_Sqrt, _ValArray, _Tp>, _Tp> sqrt (const valarray<_Tp> &__v)`
- `template<class _Dom> _Expr<_UnClos<_Sqrt, _Expr, _Dom>, typename _Dom::value_type> sqrt (const _Expr<_Dom, typename _Dom::value_type> &__e)`
- `template<typename _Blter, typename _Predicate> _Blter stable_partition (_Blter, _Blter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate> _ForwardIterator stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _RAIter> void stable_sort (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare> void stable_sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator> void stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare> void stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Tp, typename _Tp1> shared_ptr<_Tp> static_pointer_cast (const shared_ptr<_Tp1> &__r) noexcept`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp> __shared_ptr<_Tp, _Lp> static_pointer_cast (const __shared_ptr<_Tp1, _Lp> &__r) noexcept`
- `void swap (_Bit_reference __x, _Bit_reference __y) noexcept`
- `void swap (_Bit_reference __x, bool &__y) noexcept`
- `void swap (bool &__x, _Bit_reference __y) noexcept`
- `template<class _T1, class _T2> void swap (pair<_T1, _T2> &__x, pair<_T1, _T2> &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, typename _Seq> void swap (stack<_Tp, _Seq> &__x, stack<_Tp, _Seq> &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, typename _Seq> void swap (queue<_Tp, _Seq> &__x, queue<_Tp, _Seq> &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp> void swap (shared_ptr<_Tp> &__a, shared_ptr<_Tp> &__b) noexcept`
- `template<typename _Tp, typename _Dp> void swap (unique_ptr<_Tp, _Dp> &__x, unique_ptr<_Tp, _Dp> &__y) noexcept`
- `template<typename _Tp> void swap (weak_ptr<_Tp> &__a, weak_ptr<_Tp> &__b) noexcept`
- `template<typename _Tp> void swap (_Tp &__a, _Tp &__b) noexcept(__and<is_nothrow_move_constructible<_Tp>, is_nothrow_move_assignable<_Tp>>::value)`
- `template<typename _Tp, typename _Sequence, typename _Compare> void swap (priority_queue<_Tp, _Sequence, _Compare> &__x, priority_queue<_Tp, _Sequence, _Compare> &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, size_t _Nm> void swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(noexcept(swap(*__a, *__b)))`
- `template<typename _Ch_type, typename _Rx_traits> void swap (basic_regex<_Ch_type, _Rx_traits> &__lhs, basic_regex<_Ch_type, _Rx_traits> &__rhs)`
- `template<typename _Key, typename _Compare, typename _Alloc> void swap (multiset<_Key, _Compare, _Alloc> &__x, multiset<_Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc> void swap (set<_Key, _Compare, _Alloc> &__x, set<_Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> void swap (multimap<_Key, _Tp, _Compare, _Alloc> &__x, multimap<_Key, _Tp, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc> void swap (_Rb_tree<_Key, _Val, _KeyOfValue, _Compare, _Alloc> &__x, _Rb_tree<_Key, _Val, _KeyOfValue, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> void swap (map<_Key, _Tp, _Compare, _Alloc> &__x, map<_Key, _Tp, _Compare, _Alloc> &__y)`
- `template<typename _Tp, _Lock_policy _Lp> void swap (__shared_ptr<_Tp, _Lp> &__a, __shared_ptr<_Tp, _Lp> &__b) noexcept`

- `template<class _Value, class _Hash, class _Pred, class _Alloc> void swap (unordered_set< _Value, _Hash, _Pred, _Alloc> &_x, unordered_set< _Value, _Hash, _Pred, _Alloc> &_y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc> void swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc> &_x, unordered_multiset< _Value, _Hash, _Pred, _Alloc> &_y)`
- `template<typename _Tp, _Lock_policy _Lp> void swap (__weak_ptr< _Tp, _Lp> &_a, __weak_ptr< _Tp, _Lp> &_b) noexcept`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc> void swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc> &_x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc> &_y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc> void swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc> &_x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc> &_y)`
- `template<typename _Tp, typename _Alloc> void swap (forward_list< _Tp, _Alloc> &_lx, forward_list< _Tp, _Alloc> &_ly)`
- `template<typename _Tp, typename _Alloc> void swap (vector< _Tp, _Alloc> &_x, vector< _Tp, _Alloc> &_y)`
- `template<typename _Tp, typename _Alloc> void swap (list< _Tp, _Alloc> &_x, list< _Tp, _Alloc> &_y)`
- `template<typename _Bi_iter, typename _Alloc> void swap (match_results< _Bi_iter, _Alloc> &_lhs, match_results< _Bi_iter, _Alloc> &_rhs)`
- `template<typename _Tp, typename _Alloc> void swap (deque< _Tp, _Alloc> &_x, deque< _Tp, _Alloc> &_y)`
- `template<typename _CharT, typename _Traits, typename _Alloc> void swap (basic_string< _CharT, _Traits, _Alloc> &_lhs, basic_string< _CharT, _Traits, _Alloc> &_rhs)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2> _ForwardIterator2 swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _Filter1, typename _Filter2> _Filter2 swap_ranges (_Filter1, _Filter1, _Filter2)`
- `template<typename _Tp> _Expr< _UnClos< _Tan, _ValArray, _Tp>, _Tp> tan (const valarray< _Tp> &_v)`
- `template<class _Dom> _Expr< _UnClos< _Tan, _Expr, _Dom>, typename _Dom::value_type> tan (const _Expr< _Dom, typename _Dom::value_type> &_e)`
- `template<class _Dom> _Expr< _UnClos< _Tanh, _Expr, _Dom>, typename _Dom::value_type> tanh (const _Expr< _Dom, typename _Dom::value_type> &_e)`
- `template<typename _Tp> _Expr< _UnClos< _Tanh, _ValArray, _Tp>, _Tp> tanh (const valarray< _Tp> &_v)`
- `template<typename _Ex> void throw_with_nested (_Ex __ex)`
- `template<typename _CharT> _CharT tolower (_CharT __c, const locale &__loc)`
- `template<typename _CharT> _CharT toupper (_CharT __c, const locale &__loc)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation> _OIter transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BinaryOperation> _OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BinaryOperation)`
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation> _OutputIterator transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation> _OutputIterator transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _ForwardIterator> _ForwardIterator uninitialized_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator> _ForwardIterator uninitialized_copy_n (_InputIterator __first, _Size __n, _ForwardIterator __result)`
- `template<typename _ForwardIterator, typename _Tp> void uninitialized_fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &_x)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp> void uninitialized_fill_n (_ForwardIterator __first, _Size __n, const _Tp &_x)`
- `template<typename _Filter> _Filter unique (_Filter, _Filter)`
- `template<typename _Filter, typename _BinaryPredicate> _Filter unique (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _ForwardIterator> _ForwardIterator unique (_ForwardIterator __first, _ForwardIterator __last)`

- `template<typename _ForwardIterator, typename _BinaryPredicate> _ForwardIterator unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _Iiter, typename _Oiter> _Oiter unique_copy (_Iiter, _Iiter, _Oiter)`
- `template<typename _Iiter, typename _Oiter, typename _BinaryPredicate> _Oiter unique_copy (_Iiter, _Iiter, _Oiter, _BinaryPredicate)`
- `template<typename _InputIterator, typename _OutputIterator> _OutputIterator unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate> _OutputIterator unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred)`
- `ios_base & unitbuf (ios_base & __base)`
- `template<typename _Filter, typename _Tp> _Filter upper_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare> _Filter upper_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _ForwardIterator, typename _Tp> _ForwardIterator upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare> _ForwardIterator upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, _Compare __comp)`
- `ios_base & uppercase (ios_base & __base)`
- `template<typename _Facet> const _Facet & use_facet (const locale &)`

### Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits> bool regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits> bool regex_match (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Alloc, typename _Rx_traits> bool regex_match (const _Ch_type * __s, match_results< const _Ch_type *, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits> bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Rx_traits> bool regex_match (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits> bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator > & __s, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits> bool regex_search (_Bi_iter __first, _Bi_iter __last, match_results< _Bi_iter, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits> bool regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Alloc, class _Rx_traits> bool regex_search (const _Ch_type * __s, match_results< const _Ch_type *, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Rx_traits> bool regex_search (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits> bool regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > & __s, const basic_regex< _Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __flags=regex_constants::match_default)`

- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits> bool regex\_search (const basic\_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match\_results< typename basic\_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic\_regex< _Ch_type, _Rx_traits > &__e, regex\_constants::match\_flag\_type __f=regex\_constants::match\_default)`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type> _Out_iter regex\_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic\_regex< _Ch_type, _Rx_traits > &__e, const basic\_string< _Ch_type > &__fmt, regex\_constants::match\_flag\_type __flags=regex\_constants::match\_default)`
- `template<typename _Rx_traits, typename _Ch_type> basic\_string< _Ch_type > regex\_replace (const basic\_string< _Ch_type > &__s, const basic\_regex< _Ch_type, _Rx_traits > &__e, const basic\_string< _Ch_type > &__fmt, regex\_constants::match\_flag\_type __flags=regex\_constants::match\_default)`

## Variables

- `constexpr allocator\_arg\_t allocator\_arg`
- `decltype(nullptr) typedef nullptr\_t`
- `constexpr piecewise\_construct\_t piecewise\_construct`

### 3.11.1 Detailed Description

ISO C++ entities toplevel namespace is std.

### 3.11.2 Typedef Documentation

3.11.2.1 `template<bool _Cache> using std::__umap_traits = typedef __detail::__Hashtable_traits<_Cache, false, true>`

Base types for unordered\_map.

Definition at line 39 of file unordered\_map.h.

3.11.2.2 `template<bool _Cache> using std::__ummap_traits = typedef __detail::__Hashtable_traits<_Cache, false, false>`

Base types for unordered\_multimap.

Definition at line 56 of file unordered\_map.h.

3.11.2.3 `template<bool _Cache> using std::__umset_traits = typedef __detail::__Hashtable_traits<_Cache, true, false>`

Base types for unordered\_multiset.

Definition at line 54 of file unordered\_set.h.

3.11.2.4 `template<bool _Cache> using std::__uset_traits = typedef __detail::__Hashtable_traits<_Cache, true, true>`

Base types for unordered\_set.

Definition at line 39 of file unordered\_set.h.

### 3.11.2.5 typedef long long std::streamoff

Type used by fpos, char\_traits<char>, and char\_traits<wchar\_t>.

In clauses 21.1.3.1 and 27.4.1 streamoff is described as an implementation defined type. Note: In versions of GCC up to and including GCC 3.3, streamoff was typedef long.

Definition at line 94 of file postypes.h.

### 3.11.2.6 `typedef fpos<mbstate_t> std::streampos`

File position for char streams.

Definition at line 228 of file postypes.h.

### 3.11.2.7 `typedef ptrdiff_t std::streamsize`

Integral type for I/O operation counts and buffer sizes.

Definition at line 98 of file postypes.h.

### 3.11.2.8 `typedef fpos<mbstate_t> std::u16streampos`

File position for char16\_t streams.

Definition at line 234 of file postypes.h.

### 3.11.2.9 `typedef fpos<mbstate_t> std::u32streampos`

File position for char32\_t streams.

Definition at line 236 of file postypes.h.

### 3.11.2.10 `typedef fpos<mbstate_t> std::wstreampos`

File position for wchar\_t streams.

Definition at line 230 of file postypes.h.

## 3.11.3 Enumeration Type Documentation

### 3.11.3.1 anonymous enum

**Todo** Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation+style.html> This controls some aspect of the sort routines.

Definition at line 2201 of file stl\_algo.h.

## 3.11.4 Function Documentation

### 3.11.4.1 `template<typename _RandomAccessIterator > void std::__final_insertion_sort ( _RandomAccessIterator __first, _RandomAccessIterator __last )`

This is a helper function for the sort routine.

Definition at line 2206 of file stl\_algo.h.

References `__insertion_sort()`, and `__unguarded_insertion_sort()`.

Referenced by `sort()`.

### 3.11.4.2 `template<typename _RandomAccessIterator, typename _Compare > void std::__final_insertion_sort ( _RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp )`

This is a helper function for the sort routine.

Definition at line 2221 of file stl\_algo.h.

References `__insertion_sort()`, and `__unguarded_insertion_sort()`.

3.11.4.3 `template<typename _InputIterator, typename _Tp> _InputIterator std::__find ( _InputIterator __first, _InputIterator __last, const _Tp & __val, input_iterator_tag ) [inline]`

This is an overload used by `find()` for the Input Iterator case.

Definition at line 136 of file `stl_algo.h`.

Referenced by `find()`.

3.11.4.4 `template<typename _RandomAccessIterator, typename _Tp> _RandomAccessIterator std::__find ( _RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp & __val, random_access_iterator_tag )`

This is an overload used by `find()` for the RAI case.

Definition at line 158 of file `stl_algo.h`.

3.11.4.5 `template<typename _InputIterator, typename _Predicate> _InputIterator std::__find_if ( _InputIterator __first, _InputIterator __last, _Predicate __pred, input_iterator_tag ) [inline]`

This is an overload used by `find_if()` for the Input Iterator case.

Definition at line 147 of file `stl_algo.h`.

Referenced by `find_if()`.

3.11.4.6 `template<typename _RandomAccessIterator, typename _Predicate> _RandomAccessIterator std::__find_if ( _RandomAccessIterator __first, _RandomAccessIterator __last, _Predicate __pred, random_access_iterator_tag )`

This is an overload used by `find_if()` for the RAI case.

Definition at line 206 of file `stl_algo.h`.

3.11.4.7 `template<typename _InputIterator, typename _Predicate> _InputIterator std::__find_if_not ( _InputIterator __first, _InputIterator __last, _Predicate __pred, input_iterator_tag ) [inline]`

This is an overload used by `find_if_not()` for the Input Iterator case.

Definition at line 254 of file `stl_algo.h`.

Referenced by `__find_if_not()`, `find_if_not()`, and `stable_partition()`.

3.11.4.8 `template<typename _RandomAccessIterator, typename _Predicate> _RandomAccessIterator std::__find_if_not ( _RandomAccessIterator __first, _RandomAccessIterator __last, _Predicate __pred, random_access_iterator_tag )`

This is an overload used by `find_if_not()` for the RAI case.

Definition at line 265 of file `stl_algo.h`.

3.11.4.9 `template<typename _InputIterator, typename _Predicate> _InputIterator std::__find_if_not ( _InputIterator __first, _InputIterator __last, _Predicate __pred ) [inline]`

Provided for `stable_partition` to use.

Definition at line 313 of file `stl_algo.h`.

References `__find_if_not()`, and `__iterator_category()`.

3.11.4.10 `template<typename _InputIterator, typename _Predicate, typename _Distance> _InputIterator std::__find_if_not_n ( _InputIterator __first, _Distance & __len, _Predicate __pred )`

Like `find_if_not()`, but uses and updates a count of the remaining range length instead of comparing against an end iterator.

Definition at line 325 of file `stl_algo.h`.

Referenced by `__inplace_stable_partition()`, and `__stable_partition_adaptive()`.

**3.11.4.11** `template<typename _EuclideanRingElement > _EuclideanRingElement std::__gcd ( _EuclideanRingElement __m, _EuclideanRingElement __n )`

This is a helper function for the rotate algorithm specialized on RAIs. It returns the greatest common divisor of two integer values.

Definition at line 1494 of file `stl_algo.h`.

**3.11.4.12** `template<typename _RandomAccessIterator > void std::__heap_select ( _RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last )`

This is a helper function for the sort routines.

Definition at line 1929 of file `stl_algo.h`.

References `make_heap()`.

Referenced by `partial_sort()`.

**3.11.4.13** `template<typename _RandomAccessIterator, typename _Compare > void std::__heap_select ( _RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp )`

This is a helper function for the sort routines.

Definition at line 1942 of file `stl_algo.h`.

References `make_heap()`.

**3.11.4.14** `template<typename _ForwardIterator, typename _Predicate, typename _Distance > _ForwardIterator std::__inplace_stable_partition ( _ForwardIterator __first, _Predicate __pred, _Distance __len )`

This is a helper function... Requires `__len != 0` and `!__pred(*__first)`, same as `__stable_partition_adaptive`.

Definition at line 1785 of file `stl_algo.h`.

References `__find_if_not_n()`, `advance()`, `distance()`, and `rotate()`.

Referenced by `stable_partition()`.

**3.11.4.15** `template<typename _RandomAccessIterator > void std::__inplace_stable_sort ( _RandomAccessIterator __first, _RandomAccessIterator __last )`

This is a helper function for the stable sorting routines.

Definition at line 3486 of file `stl_algo.h`.

References `__insertion_sort()`, and `__merge_without_buffer()`.

Referenced by `__inplace_stable_sort()`, and `stable_sort()`.

**3.11.4.16** `template<typename _RandomAccessIterator, typename _Compare > void std::__inplace_stable_sort ( _RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp )`

This is a helper function for the stable sorting routines.

Definition at line 3505 of file `stl_algo.h`.

References `__inplace_stable_sort()`, `__insertion_sort()`, and `__merge_without_buffer()`.



3.11.4.17 `template<typename _RandomAccessIterator > void std::__insertion_sort ( _RandomAccessIterator __first, _RandomAccessIterator __last )`

This is a helper function for the sort routine.

Definition at line 2129 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

Referenced by `__final_insertion_sort()`, and `__inplace_stable_sort()`.

3.11.4.18 `template<typename _RandomAccessIterator, typename _Compare > void std::__insertion_sort ( _RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp )`

This is a helper function for the sort routine.

Definition at line 2152 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

3.11.4.19 `template<typename _RandomAccessIterator, typename _Size > void std::__introsort_loop ( _RandomAccessIterator __first, _RandomAccessIterator __last, _Size __depth_limit )`

This is a helper function for the sort routine.

Definition at line 2302 of file `stl_algo.h`.

References `__unguarded_partition_pivot()`, and `partial_sort()`.

Referenced by `__introsort_loop()`, and `sort()`.

3.11.4.20 `template<typename _RandomAccessIterator, typename _Size, typename _Compare > void std::__introsort_loop ( _RandomAccessIterator __first, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp )`

This is a helper function for the sort routine.

Definition at line 2324 of file `stl_algo.h`.

References `__introsort_loop()`, `__unguarded_partition_pivot()`, and `partial_sort()`.

3.11.4.21 `constexpr int std::__lg ( int __n ) [inline]`

This is a helper function for the sort routines and for `random.tcc`.

Definition at line 980 of file `stl_algobase.h`.

Referenced by `nth_element()`, and `sort()`.

3.11.4.22 `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer > void std::__merge_adaptive ( _BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size )`

This is a helper function for the merge routines.

Definition at line 2934 of file `stl_algo.h`.

References `__move_merge_adaptive()`, `__move_merge_adaptive_backward()`, `__rotate_adaptive()`, `advance()`, `distance()`, `lower_bound()`, and `upper_bound()`.

Referenced by `__merge_adaptive()`, and `inplace_merge()`.

3.11.4.23 `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _Compare> void std::__merge_adaptive ( _BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Compare __comp )`

This is a helper function for the merge routines.

Definition at line 2990 of file `stl_algo.h`.

References `__merge_adaptive()`, `__move_merge_adaptive()`, `__move_merge_adaptive_backward()`, `__rotate_adaptive()`, `advance()`, `distance()`, `lower_bound()`, and `upper_bound()`.

3.11.4.24 `template<typename _BidirectionalIterator, typename _Distance> void std::__merge_without_buffer ( _BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2 )`

This is a helper function for the merge routines.

Definition at line 3047 of file `stl_algo.h`.

References `advance()`, `distance()`, `iter_swap()`, `lower_bound()`, `rotate()`, and `upper_bound()`.

Referenced by `__inplace_stable_sort()`, `__merge_without_buffer()`, and `inplace_merge()`.

3.11.4.25 `template<typename _BidirectionalIterator, typename _Distance, typename _Compare> void std::__merge_without_buffer ( _BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Compare __comp )`

This is a helper function for the merge routines.

Definition at line 3091 of file `stl_algo.h`.

References `__merge_without_buffer()`, `advance()`, `distance()`, `iter_swap()`, `lower_bound()`, `rotate()`, and `upper_bound()`.

3.11.4.26 `template<typename _Iterator> void std::__move_median_to_first ( _Iterator __result, _Iterator __a, _Iterator __b, _Iterator __c )`

Swaps the median value of `*__a`, `*__b` and `*__c` to `*__result`.

Definition at line 78 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__unguarded_partition_pivot()`.

3.11.4.27 `template<typename _Iterator, typename _Compare> void std::__move_median_to_first ( _Iterator __result, _Iterator __a, _Iterator __b, _Iterator __c, _Compare __comp )`

Swaps the median value of `*__a`, `*__b` and `*__c` under `__comp` to `*__result`.

Definition at line 105 of file `stl_algo.h`.

References `iter_swap()`.

3.11.4.28 `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator> _OutputIterator std::__move_merge ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result )`

This is a helper function for the `__merge_sort_loop` routines.

Definition at line 3249 of file `stl_algo.h`.

3.11.4.29 `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare  
> _OutputIterator std::__move_merge ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,  
_InputIterator2 __last2, _OutputIterator __result, _Compare __comp )`

This is a helper function for the `__merge_sort_loop` routines.

Definition at line 3276 of file `stl_algo.h`.

3.11.4.30 `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator > void  
std::__move_merge_adaptive ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2  
__last2, _OutputIterator __result )`

This is a helper function for the `__merge_adaptive` routines.

Definition at line 2755 of file `stl_algo.h`.

Referenced by `__merge_adaptive()`.

3.11.4.31 `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare > void  
std::__move_merge_adaptive ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2  
__last2, _OutputIterator __result, _Compare __comp )`

This is a helper function for the `__merge_adaptive` routines.

Definition at line 2781 of file `stl_algo.h`.

3.11.4.32 `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BidirectionalIterator3  
> void std::__move_merge_adaptive_backward ( _BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1,  
_BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result )`

This is a helper function for the `__merge_adaptive` routines.

Definition at line 2807 of file `stl_algo.h`.

Referenced by `__merge_adaptive()`.

3.11.4.33 `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BidirectionalIterator3  
, typename _Compare > void std::__move_merge_adaptive_backward ( _BidirectionalIterator1 __first1,  
_BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3  
__result, _Compare __comp )`

This is a helper function for the `__merge_adaptive` routines.

Definition at line 2849 of file `stl_algo.h`.

3.11.4.34 `template<typename _ForwardIterator, typename _Predicate > _ForwardIterator std::__partition ( _ForwardIterator __first,  
_ForwardIterator __last, _Predicate __pred, forward_iterator_tag )`

This is a helper function...

Definition at line 1728 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `partition()`.

3.11.4.35 `template<typename _BidirectionalIterator, typename _Predicate > _BidirectionalIterator std::__partition (   
_BidirectionalIterator __first, _BidirectionalIterator __last, _Predicate __pred, bidirectional_iterator_tag )`

This is a helper function...

Definition at line 1753 of file `stl_algo.h`.

References `iter_swap()`.

**3.11.4.36** `template<typename _BidirectionalIterator > void std::__reverse ( _BidirectionalIterator __first, _BidirectionalIterator __last, bidirectional_iterator_tag )`

This is an uglified `reverse(_BidirectionalIterator, _BidirectionalIterator)` overloaded for bidirectional iterators.

Definition at line 1394 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__rotate()`, and `reverse()`.

**3.11.4.37** `template<typename _RandomAccessIterator > void std::__reverse ( _RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag )`

This is an uglified `reverse(_BidirectionalIterator, _BidirectionalIterator)` overloaded for random access iterators.

Definition at line 1414 of file `stl_algo.h`.

References `iter_swap()`.

**3.11.4.38** `template<typename _ForwardIterator > void std::__rotate ( _ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, forward_iterator_tag )`

This is a helper function for the rotate algorithm.

Definition at line 1508 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__gnu_cxx::bitmap_allocator< _Tp >::M_deallocate_single_object()`, and `rotate()`.

**3.11.4.39** `template<typename _BidirectionalIterator > void std::__rotate ( _BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, bidirectional_iterator_tag )`

This is a helper function for the rotate algorithm.

Definition at line 1544 of file `stl_algo.h`.

References `__reverse()`, and `iter_swap()`.

**3.11.4.40** `template<typename _RandomAccessIterator > void std::__rotate ( _RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, random_access_iterator_tag )`

This is a helper function for the rotate algorithm.

Definition at line 1574 of file `stl_algo.h`.

References `iter_swap()`, `swap()`, and `swap_ranges()`.

**3.11.4.41** `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _Distance > _BidirectionalIterator1 std::__rotate_adaptive ( _BidirectionalIterator1 __first, _BidirectionalIterator1 __middle, _BidirectionalIterator1 __last, _Distance __len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance __buffer_size )`

This is a helper function for the merge routines.

Definition at line 2892 of file `stl_algo.h`.

References `advance()`, `distance()`, and `rotate()`.

Referenced by `__merge_adaptive()`.

3.11.4.42 `template<typename _ForwardIterator, typename _Integer, typename _Tp > _ForwardIterator std::__search_n (`  
`_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp & __val, std::forward_iterator_tag )`

This is an uglified `search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&)` overloaded for forward iterators.

Definition at line 353 of file `stl_algo.h`.

References `find()`.

Referenced by `search_n()`.

3.11.4.43 `template<typename _RandomAccessIter, typename _Integer, typename _Tp > _RandomAccessIter`  
`std::__search_n ( _RandomAccessIter __first, _RandomAccessIter __last, _Integer __count, const _Tp & __val,`  
`std::random_access_iterator_tag )`

This is an uglified `search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&)` overloaded for random access iterators.

Definition at line 385 of file `stl_algo.h`.

3.11.4.44 `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_ForwardIterator std::__search_n ( _ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp & __val,`  
`_BinaryPredicate __binary_pred, std::forward_iterator_tag )`

This is an uglified `search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&, _BinaryPredicate)` overloaded for forward iterators.

Definition at line 424 of file `stl_algo.h`.

3.11.4.45 `template<typename _RandomAccessIter, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_RandomAccessIter std::__search_n ( _RandomAccessIter __first, _RandomAccessIter __last, _Integer __count, const`  
`_Tp & __val, _BinaryPredicate __binary_pred, std::random_access_iterator_tag )`

This is an uglified `search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&, _BinaryPredicate)` overloaded for random access iterators.

Definition at line 463 of file `stl_algo.h`.

3.11.4.46 `template<typename _ForwardIterator, typename _Pointer, typename _Predicate, typename _Distance > _ForwardIterator`  
`std::__stable_partition_adaptive ( _ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, _Distance __len,`  
`_Pointer __buffer, _Distance __buffer_size )`

This is a helper function... Requires `__first != __last` and `!__pred(*__first)` and `__len == distance(__first, __last)`.

`!__pred(*__first)` allows us to guarantee that we don't move-assign an element onto itself.

Definition at line 1817 of file `stl_algo.h`.

References `__find_if_not_n()`, `advance()`, `distance()`, and `rotate()`.

Referenced by `stable_partition()`.

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

This is a helper function for the sort routine.

Definition at line 2174 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

Referenced by `__final_insertion_sort()`.

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

This is a helper function for the sort routine.

Definition at line 2187 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

3.11.4.49 `template<typename _RandomAccessIterator> void std::__unguarded_linear_insert ( _RandomAccessIterator __last )`

This is a helper function for the sort routine.

Definition at line 2092 of file `stl_algo.h`.

Referenced by `__insertion_sort()`, and `__unguarded_insertion_sort()`.

3.11.4.50 `template<typename _RandomAccessIterator, typename _Compare> void std::__unguarded_linear_insert ( _RandomAccessIterator __last, _Compare __comp )`

This is a helper function for the sort routine.

Definition at line 2110 of file `stl_algo.h`.

3.11.4.51 `template<typename _RandomAccessIterator, typename _Tp> _RandomAccessIterator std::__unguarded_partition ( _RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp & __pivot )`

This is a helper function...

Definition at line 2237 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__unguarded_partition_pivot()`.

3.11.4.52 `template<typename _RandomAccessIterator, typename _Tp, typename _Compare> _RandomAccessIterator std::__unguarded_partition ( _RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp & __pivot, _Compare __comp )`

This is a helper function...

Definition at line 2257 of file `stl_algo.h`.

References `iter_swap()`.

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

This is a helper function...

Definition at line 2278 of file `stl_algo.h`.

References `__move_median_to_first()`, and `__unguarded_partition()`.

Referenced by `__introsort_loop()`.

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

This is a helper function...

Definition at line 2290 of file `stl_algo.h`.

References `__move_median_to_first()`, and `__unguarded_partition()`.

**3.11.4.55** `template<typename _ForwardIterator, typename _OutputIterator > _OutputIterator std::__unique_copy ( _ForwardIterator __first, _ForwardIterator __last, _OutputIterator __result, forward_iterator_tag, output_iterator_tag )`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator)` overloaded for forward iterators and output iterator as result.

Definition at line 1246 of file `stl_algo.h`.

Referenced by `unique_copy()`.

**3.11.4.56** `template<typename _InputIterator, typename _OutputIterator > _OutputIterator std::__unique_copy ( _InputIterator __first, _InputIterator __last, _OutputIterator __result, input_iterator_tag, output_iterator_tag )`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator)` overloaded for input iterators and output iterator as result.

Definition at line 1269 of file `stl_algo.h`.

**3.11.4.57** `template<typename _InputIterator, typename _ForwardIterator > _ForwardIterator std::__unique_copy ( _InputIterator __first, _InputIterator __last, _ForwardIterator __result, input_iterator_tag, forward_iterator_tag )`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator)` overloaded for input iterators and forward iterator as result.

Definition at line 1292 of file `stl_algo.h`.

**3.11.4.58** `template<typename _ForwardIterator, typename _OutputIterator, typename _BinaryPredicate > _OutputIterator std::__unique_copy ( _ForwardIterator __first, _ForwardIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, forward_iterator_tag, output_iterator_tag )`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for forward iterators and output iterator as result.

Definition at line 1313 of file `stl_algo.h`.

**3.11.4.59** `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate > _OutputIterator std::__unique_copy ( _InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag, output_iterator_tag )`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for input iterators and output iterator as result.

Definition at line 1342 of file `stl_algo.h`.

**3.11.4.60** `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate > _ForwardIterator std::__unique_copy ( _InputIterator __first, _InputIterator __last, _ForwardIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag, forward_iterator_tag )`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for input iterators and forward iterator as result.

Definition at line 1371 of file `stl_algo.h`.

**3.11.4.61** `template<typename _T1, typename... _Args> void std::__Construct ( _T1 * __p, _Args &&... __args ) [inline]`

Constructs an object in existing memory by invoking an allocated object's constructor with an initializer.

Definition at line 74 of file `stl_construct.h`.

3.11.4.62 `template<typename _Tp> void std::_Destroy ( _Tp * __pointer ) [inline]`

Destroy the object pointed to by a pointer type.

Definition at line 92 of file `stl_construct.h`.

Referenced by `std::vector< _State >::~~vector()`.

3.11.4.63 `template<typename _ForwardIterator> void std::_Destroy ( _ForwardIterator __first, _ForwardIterator __last ) [inline]`

Destroy a range of objects. If the `value_type` of the object has a trivial destructor, the compiler should optimize all of this away, otherwise the objects' destructors must be invoked.

Definition at line 122 of file `stl_construct.h`.

3.11.4.64 `template<typename _ForwardIterator, typename _Allocator> void std::_Destroy ( _ForwardIterator __first, _ForwardIterator __last, _Allocator & __alloc )`

Destroy a range of objects using the supplied allocator. For nondefault allocators we do not optimize away invocation of `destroy()` even if `_Tp` has a trivial destructor.

Definition at line 138 of file `stl_construct.h`.

References `__addressof()`.

3.11.4.65 `template<typename _InputIterator, typename _Tp> _Tp std::accumulate ( _InputIterator __first, _InputIterator __last, _Tp __init ) [inline]`

Accumulate values in a range.

Accumulates the values in the range `[first,last)` using `operator+()`. The initial value is `init`. The values are processed in order.

#### Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__init</code>	Starting value to add other values to.

#### Returns

The final sum.

Definition at line 120 of file `stl_numeric.h`.

Referenced by `__gnu_parallel::__parallel_partial_sum_linear()`.

3.11.4.66 `template<typename _InputIterator, typename _Tp, typename _BinaryOperation> _Tp std::accumulate ( _InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op ) [inline]`

Accumulate values in a range with operation.

Accumulates the values in the range `[first,last)` using the function object `__binary_op`. The initial value is `__init`. The values are processed in order.



**Parameters**

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__init</code>	Starting value to add other values to.
<code>__binary_op</code>	Function object to accumulate with.

**Returns**

The final sum.

Definition at line 146 of file `stl_numeric.h`.

**3.11.4.67** `template<typename _InputIterator, typename _OutputIterator > _OutputIterator std::adjacent_difference ( _InputIterator __first, _InputIterator __last, _OutputIterator __result )`

Return differences between adjacent values.

Computes the difference between adjacent values in the range `[first,last)` using `operator-()` and writes the result to `__result`.

**Parameters**

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Output sums.

**Returns**

Iterator pointing just beyond the values written to result.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 539. `partial_sum` and `adjacent_difference` should mention requirements

Definition at line 317 of file `stl_numeric.h`.

**3.11.4.68** `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation > _OutputIterator std::adjacent_difference ( _InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op )`

Return differences between adjacent values.

Computes the difference between adjacent values in the range `[__first,__last)` using the function object `__binary_op` and writes the result to `__result`.

**Parameters**

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Output sum.
<code>__binary_op</code>	Function object.

**Returns**

Iterator pointing just beyond the values written to result.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 539. `partial_sum` and `adjacent_difference` should mention requirements

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

```
3.11.4.69  template<typename _InputIterator, typename _Distance> void std::advance ( _InputIterator & __i, _Distance __n )  
           [inline]
```

A generalization of pointer arithmetic.

## Parameters

<code>__i</code>	An input iterator.
<code>__n</code>	The <i>delta</i> by which to change <code>__i</code> .

## Returns

Nothing.

This increments `i` by `n`. For bidirectional and random access iterators, `__n` may be negative, in which case `__i` is decremented.

For random access iterators, this uses their `+` and `-` operations and are constant time. For other iterator classes they are linear time.

Definition at line 173 of file `stl_iterator_base_funcs.h`.

References `__iterator_category()`.

Referenced by `__inplace_stable_partition()`, `__merge_adaptive()`, `__merge_without_buffer()`, `__rotate_adaptive()`, `__stable_partition_adaptive()`, `equal_range()`, `__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::get_child()`, `__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::get_child()`, `is_permutation()`, `lower_bound()`, `partition_point()`, and `upper_bound()`.

**3.11.4.70** `template<class _Container > auto std::begin ( _Container & __cont )-> decltype(__cont.begin())` `[inline]`

Return an iterator pointing to the first element of the container.

## Parameters

<code>__cont</code>	Container.
---------------------	------------

Definition at line 48 of file `range_access.h`.

Referenced by `__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::get_child()`.

**3.11.4.71** `template<class _Container > auto std::begin ( const _Container & __cont )-> decltype(__cont.begin())` `[inline]`

Return an iterator pointing to the first element of the const container.

## Parameters

<code>__cont</code>	Container.
---------------------	------------

Definition at line 58 of file `range_access.h`.

**3.11.4.72** `template<class _Tp, size_t _Nm> _Tp* std::begin ( _Tp(&) __arr[_Nm] )` `[inline]`

Return an iterator pointing to the first element of the array.

## Parameters

<code>__arr</code>	Array.
--------------------	--------

Definition at line 87 of file `range_access.h`.

**3.11.4.73** `ios_base& std::boolalpha ( ios_base & __base )` `[inline]`

Calls `base.setf(ios_base::boolalpha)`.

Definition at line 795 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::boolalpha`, and `std::ios_base::setf()`.

**3.11.4.74** `template<typename _Tp, typename _Tp1, _Lock_policy _Lp> __shared_ptr<_Tp, _Lp> std::const_pointer_cast ( const __shared_ptr<_Tp1, _Lp> &__r ) [inline], [noexcept]`

`const_pointer_cast`

Definition at line 1167 of file `shared_ptr_base.h`.

**3.11.4.75** `ios_base& std::dec ( ios_base &__base ) [inline]`

Calls `base.setf(ios_base::dec, ios_base::basefield)`.

Definition at line 933 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::basefield`, `std::ios_base::dec`, and `std::ios_base::setf()`.

**3.11.4.76** `template<typename _InputIterator> iterator_traits<_InputIterator>::difference_type std::distance ( _InputIterator __first, _InputIterator __last ) [inline]`

A generalization of pointer arithmetic.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

**Returns**

The distance between them.

Returns `n` such that `__first + n == __last`. This requires that `__last` must be reachable from `__first`. Note that `n` may be negative.

For random access iterators, this uses their `+` and `-` operations and are constant time. For other iterator classes they are linear time.

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

References `__iterator_category()`.

Referenced by `__inplace_stable_partition()`, `__merge_adaptive()`, `__merge_without_buffer()`, `__rotate_adaptive()`, `__stable_partition_adaptive()`, `equal_range()`, `inplace_merge()`, `is_heap_until()`, `is_permutation()`, `std::sub_match<__Bi_iter>::length()`, `lower_bound()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, `__gnu_parallel::detail::pat_trie_base::Node_citer<Node, Leaf, Head, Inode, _Clterator, Iterator, _Alloc>::num_children()`, `partition_point()`, `std::match_results<_FwdIterT, _Alloc>::position()`, `std::list<__inp, __rebind_inp>::size()`, and `upper_bound()`.

**3.11.4.77** `template<typename _Tp, typename _Tp1, _Lock_policy _Lp> __shared_ptr<_Tp, _Lp> std::dynamic_pointer_cast ( const __shared_ptr<_Tp1, _Lp> &__r ) [inline], [noexcept]`

`dynamic_pointer_cast`

Definition at line 1177 of file `shared_ptr_base.h`.

**3.11.4.78** `template<class _Container> auto std::end ( _Container &__cont )-> decltype(__cont.end()) [inline]`

Return an iterator pointing to one past the last element of the container.

## Parameters

<code>__cont</code>	Container.
---------------------	------------

Definition at line 68 of file `range_access.h`.

**3.11.4.79** `template<class _Container > auto std::end ( const _Container & __cont ) -> decltype(__cont.end())` `[inline]`

Return an iterator pointing to one past the last element of the const container.

## Parameters

<code>__cont</code>	Container.
---------------------	------------

Definition at line 78 of file `range_access.h`.

**3.11.4.80** `template<class _Tp, size_t _Nm> _Tp* std::end ( _Tp(&) __arr[_Nm] )` `[inline]`

Return an iterator pointing to one past the last element of the array.

## Parameters

<code>__arr</code>	Array.
--------------------	--------

Definition at line 97 of file `range_access.h`.

**3.11.4.81** `ios_base& std::fixed ( ios_base & __base )` `[inline]`

Calls `base.setf(ios_base::fixed, ios_base::floatfield)`.

Definition at line 958 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::fixed`, `std::ios_base::floatfield`, and `std::ios_base::setf()`.

**3.11.4.82** `template<typename _Tp > pair<_Tp*, ptrdiff_t> std::get_temporary_buffer ( ptrdiff_t __len )` `[noexcept]`

Allocates a temporary buffer.

## Parameters

<code>__len</code>	The number of objects of type <code>Tp</code> .
--------------------	---

## Returns

See full description.

Reinventing the wheel, but this time with prettier spokes!

This function tries to obtain storage for `__len` adjacent `Tp` objects. The objects themselves are not constructed, of course. A `pair<>` is returned containing *the buffer's address and capacity (in the units of `sizeof(_Tp)`)*, or a pair of 0 values if no storage can be obtained. Note that the capacity obtained may be less than that requested if the memory is unavailable; you should compare `len` with the `.second` return value.

Provides the nothrow exception guarantee.

Definition at line 85 of file `stl_tempbuf.h`.

Referenced by `std::_Temporary_buffer<_ForwardIterator, _Tp>::_Temporary_buffer()`.

3.11.4.83 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename  
> class _Base> basic_istream<_CharT, _Traits>& std::getline ( basic_istream<_CharT, _Traits> & __is,  
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base> & __str, _CharT __delim )`

Read a line from stream into a string.

## Parameters

<code>__is</code>	Input stream.
<code>__str</code>	Buffer to store into.
<code>__delim</code>	Character marking end of line.

## Returns

Reference to the input stream.

Stores characters from `__is` into `__str` until `__delim` is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased. If `delim` was encountered, it is extracted but not stored into `__str`.

```
3.11.4.84  template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename
> class _Base> basic_istream<_CharT, _Traits>& std::getline ( basic_istream<_CharT, _Traits > & __is,
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base > & __str ) [inline]
```

Read a line from stream into a string.

## Parameters

<code>__is</code>	Input stream.
<code>__str</code>	Buffer to store into.

## Returns

Reference to the input stream.

Stores characters from `is` into `__str` until '

' is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased. If end of line was encountered, it is extracted but not stored into `__str`.

Definition at line 2567 of file `vstring.h`.

References `getline()`.

```
3.11.4.85  template<typename _CharT , typename _Traits , typename _Alloc > basic_istream<_CharT, _Traits>& std::getline (
basic_istream<_CharT, _Traits > & __is, basic_string<_CharT, _Traits, _Alloc > & __str, _CharT __delim )
```

Read a line from stream into a string.

## Parameters

<code>__is</code>	Input stream.
<code>__str</code>	Buffer to store into.
<code>__delim</code>	Character marking end of line.

## Returns

Reference to the input stream.

Stores characters from `__is` into `__str` until `__delim` is found, the end of the stream is encountered, or `str.max_size()` is reached. Any previous contents of `__str` are erased. If `__delim` is encountered, it is extracted but not stored into `__str`.

Referenced by `getline()`.

3.11.4.86 `template<typename _CharT, typename _Traits, typename _Alloc> basic_istream<_CharT, _Traits>& std::getline (`  
`basic_istream<_CharT, _Traits> & __is, basic_string<_CharT, _Traits, _Alloc> & __str ) [inline]`

Read a line from stream into a string.



## Parameters

<code>__is</code>	Input stream.
<code>__str</code>	Buffer to store into.

## Returns

Reference to the input stream.

Stores characters from `is` into `__str` until '`'` is found, the end of the stream is encountered, or `str.max_size()` is reached. Any previous contents of `__str` are erased. If end of line is encountered, it is extracted but not stored into `__str`.

Definition at line 2793 of file `basic_string.h`.

References `getline()`.

### 3.11.4.87 `ios_base& std::hex ( ios_base & __base ) [inline]`

Calls `base.setf(ios_base::hex, ios_base::basefield)`.

Definition at line 941 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::basefield`, `std::ios_base::hex`, and `std::ios_base::setf()`.

Referenced by `std::regex_traits< _Ch_type >::value()`.

### 3.11.4.88 `template<typename _InputIterator1, typename _InputIterator2, typename _Tp > _Tp std::inner_product ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init ) [inline]`

Compute inner product of two ranges.

Starting with an initial value of `__init`, multiplies successive elements from the two ranges and adds each product into the accumulated value using `operator+()`. The values in the ranges are processed in order.

## Parameters

<code>__first1</code>	Start of range 1.
<code>__last1</code>	End of range 1.
<code>__first2</code>	Start of range 2.
<code>__init</code>	Starting value to add other values to.

## Returns

The final inner product.

Definition at line 174 of file `stl_numeric.h`.

### 3.11.4.89 `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2 > _Tp std::inner_product ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2 ) [inline]`

Compute inner product of two ranges.

Starting with an initial value of `__init`, applies `__binary_op2` to successive elements from the two ranges and accumulates each result into the accumulated value using `__binary_op1`. The values in the ranges are processed in order.

## Parameters

<code>__first1</code>	Start of range 1.
<code>__last1</code>	End of range 1.
<code>__first2</code>	Start of range 2.
<code>__init</code>	Starting value to add other values to.
<code>__binary_op1</code>	Function object to accumulate with.
<code>__binary_op2</code>	Function object to apply to pairs of input values.

## Returns

The final inner product.

Definition at line 206 of file `std_numeric.h`.

**3.11.4.90** `ios_base& std::internal ( ios_base & __base ) [inline]`

Calls `base.setf(ios_base::internal, ios_base::adjustfield)`.

Definition at line 908 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::adjustfield`, and `std::ios_base::internal`.

**3.11.4.91** `template<typename _ForwardIterator, typename _Tp> void std::iota ( _ForwardIterator __first, _ForwardIterator __last, _Tp __value )`

Create a range of sequentially increasing values.

For each element in the range `[first,last)` assigns `value` and increments `value` as if by `++value`.

## Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__value</code>	Starting value.

## Returns

Nothing.

Definition at line 82 of file `std_numeric.h`.

**3.11.4.92** `template<typename _CharT> bool std::isalnum ( _CharT __c, const locale & __loc ) [inline]`

Convenience interface to `ctype.is(ctype_base::alnum, __c)`.

Definition at line 2584 of file `locale_facets.h`.

**3.11.4.93** `template<typename _CharT> bool std::isalpha ( _CharT __c, const locale & __loc ) [inline]`

Convenience interface to `ctype.is(ctype_base::alpha, __c)`.

Definition at line 2560 of file `locale_facets.h`.

**3.11.4.94** `template<typename _CharT> bool std::isctrl ( _CharT __c, const locale & __loc ) [inline]`

Convenience interface to `ctype.is(ctype_base::cntrl, __c)`.

Definition at line 2542 of file `locale_facets.h`.

**3.11.4.95** `template<typename _CharT> bool std::isdigit ( _CharT __c, const locale & __loc ) [inline]`

Convenience interface to `ctype.is(ctype_base::digit, __c)`.

Definition at line 2566 of file `locale_facets.h`.

**3.11.4.96** `template<typename _CharT> bool std::isgraph ( _CharT __c, const locale & __loc ) [inline]`

Convenience interface to `ctype.is(ctype_base::graph, __c)`.

Definition at line 2590 of file `locale_facets.h`.

**3.11.4.97** `template<typename _CharT> bool std::islower ( _CharT __c, const locale & __loc ) [inline]`

Convenience interface to `ctype.is(ctype_base::lower, __c)`.

Definition at line 2554 of file `locale_facets.h`.

**3.11.4.98** `template<typename _CharT> bool std::isprint ( _CharT __c, const locale & __loc ) [inline]`

Convenience interface to `ctype.is(ctype_base::print, __c)`.

Definition at line 2536 of file `locale_facets.h`.

**3.11.4.99** `template<typename _CharT> bool std::ispunct ( _CharT __c, const locale & __loc ) [inline]`

Convenience interface to `ctype.is(ctype_base::punct, __c)`.

Definition at line 2572 of file `locale_facets.h`.

**3.11.4.100** `template<typename _CharT> bool std::isspace ( _CharT __c, const locale & __loc ) [inline]`

Convenience interface to `ctype.is(ctype_base::space, __c)`.

Definition at line 2530 of file `locale_facets.h`.

**3.11.4.101** `template<typename _CharT> bool std::isupper ( _CharT __c, const locale & __loc ) [inline]`

Convenience interface to `ctype.is(ctype_base::upper, __c)`.

Definition at line 2548 of file `locale_facets.h`.

**3.11.4.102** `template<typename _CharT> bool std::isxdigit ( _CharT __c, const locale & __loc ) [inline]`

Convenience interface to `ctype.is(ctype_base::xdigit, __c)`.

Definition at line 2578 of file `locale_facets.h`.

**3.11.4.103** `ios_base& std::left ( ios_base & __base ) [inline]`

Calls `base.setf(ios_base::left, ios_base::adjustfield)`.

Definition at line 916 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::adjustfield`, `std::ios_base::left`, and `std::ios_base::setf()`.

**3.11.4.104** `ios_base& std::noboolalpha ( ios_base & __base ) [inline]`

Calls `base.unsetf(ios_base::boolalpha)`.

Definition at line 803 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::boolalpha`, and `std::ios_base::unsetf()`.

**3.11.4.105 ios\_base& std::noshowbase ( ios\_base & \_\_base ) [inline]**

Calls base.unsetf(ios\_base::showbase).

Definition at line 819 of file ios\_base.h.

References `__gnu_debug::__base()`, `std::ios_base::showbase`, and `std::ios_base::unsetf()`.

**3.11.4.106 ios\_base& std::noshowpoint ( ios\_base & \_\_base ) [inline]**

Calls base.unsetf(ios\_base::showpoint).

Definition at line 835 of file ios\_base.h.

References `__gnu_debug::__base()`, `std::ios_base::showpoint`, and `std::ios_base::unsetf()`.

**3.11.4.107 ios\_base& std::noshowpos ( ios\_base & \_\_base ) [inline]**

Calls base.unsetf(ios\_base::showpos).

Definition at line 851 of file ios\_base.h.

References `__gnu_debug::__base()`, `std::ios_base::showpos`, and `std::ios_base::unsetf()`.

**3.11.4.108 ios\_base& std::noskipws ( ios\_base & \_\_base ) [inline]**

Calls base.unsetf(ios\_base::skipws).

Definition at line 867 of file ios\_base.h.

References `__gnu_debug::__base()`, `std::ios_base::skipws`, and `std::ios_base::unsetf()`.

**3.11.4.109 ios\_base& std::nounitbuf ( ios\_base & \_\_base ) [inline]**

Calls base.unsetf(ios\_base::unitbuf).

Definition at line 899 of file ios\_base.h.

References `__gnu_debug::__base()`, `std::ios_base::unitbuf`, and `std::ios_base::unsetf()`.

**3.11.4.110 ios\_base& std::nouppercase ( ios\_base & \_\_base ) [inline]**

Calls base.unsetf(ios\_base::uppercase).

Definition at line 883 of file ios\_base.h.

References `__gnu_debug::__base()`, `std::ios_base::unsetf()`, and `std::ios_base::uppercase`.

**3.11.4.111 ios\_base& std::oct ( ios\_base & \_\_base ) [inline]**

Calls base.setf(ios\_base::oct, ios\_base::basefield).

Definition at line 949 of file ios\_base.h.

References `__gnu_debug::__base()`, `std::ios_base::basefield`, `std::ios_base::oct`, and `std::ios_base::setf()`.

Referenced by `std::regex_traits<_Ch_type>::value()`.

**3.11.4.112 template<typename \_Tp> bool std::operator!= ( const \_Fwd\_list\_iterator<\_Tp> & \_\_x, const \_Fwd\_list\_const\_iterator<\_Tp> & \_\_y ) [inline]**

Forward list iterator inequality comparison.

Definition at line 267 of file forward\_list.h.

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

Based on operator==.

Definition at line 267 of file `stl_stack.h`.

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

Based on operator==.

Definition at line 292 of file `stl_queue.h`.

3.11.4.115 `template<typename _Key, typename _Compare, typename _Alloc> bool std::operator!=( const multiset<_Key, _Compare, _Alloc> &__x, const multiset<_Key, _Compare, _Alloc> &__y ) [inline]`

Returns `!(x == y)`.

Definition at line 763 of file `stl_multiset.h`.

3.11.4.116 `template<typename _Key, typename _Compare, typename _Alloc> bool std::operator!=( const set<_Key, _Compare, _Alloc> &__x, const set<_Key, _Compare, _Alloc> &__y ) [inline]`

Returns `!(x == y)`.

Definition at line 778 of file `stl_set.h`.

3.11.4.117 `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool std::operator!=( const multimap<_Key, _Tp, _Compare, _Alloc> &__x, const multimap<_Key, _Tp, _Compare, _Alloc> &__y ) [inline]`

Based on operator==.

Definition at line 888 of file `stl_multimap.h`.

3.11.4.118 `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool std::operator!=( const map<_Key, _Tp, _Compare, _Alloc> &__x, const map<_Key, _Tp, _Compare, _Alloc> &__y ) [inline]`

Based on operator==.

Definition at line 986 of file `stl_map.h`.

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

Based on operator==.

Definition at line 1367 of file `forward_list.h`.

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

Based on operator==.

Definition at line 1428 of file `stl_vector.h`.

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

Based on operator==.

Definition at line 1638 of file `stl_list.h`.

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

Based on `operator==`.

Definition at line 1983 of file `stl_deque.h`.

3.11.4.123 `template<typename _CharT, typename _Traits, typename _Alloc > bool std::operator!=( const basic_string<_CharT, _Traits, _Alloc > &__lhs, const basic_string<_CharT, _Traits, _Alloc > &__rhs ) [inline]`

Test difference of two strings.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Returns

True if `__lhs.compare(__rhs) != 0`. False otherwise.

Definition at line 2532 of file `basic_string.h`.

3.11.4.124 `template<typename _CharT, typename _Traits, typename _Alloc > bool std::operator!=( const _CharT * __lhs, const basic_string<_CharT, _Traits, _Alloc > &__rhs ) [inline]`

Test difference of C string and string.

Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

Returns

True if `__rhs.compare(__lhs) != 0`. False otherwise.

Definition at line 2544 of file `basic_string.h`.

3.11.4.125 `template<typename _CharT, typename _Traits, typename _Alloc > bool std::operator!=( const basic_string<_CharT, _Traits, _Alloc > &__lhs, const _CharT * __rhs ) [inline]`

Test difference of string and C string.

Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

Returns

True if `__lhs.compare(__rhs) != 0`. False otherwise.

Definition at line 2556 of file `basic_string.h`.

3.11.4.126 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string<_CharT, _Traits, _Alloc>  
std::operator+ ( const basic_string< _CharT, _Traits, _Alloc> & __lhs, const basic_string< _CharT, _Traits,  
_Alloc> & __rhs )`

Concatenate two strings.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

## Returns

New string with value of `__lhs` followed by `__rhs`.

Definition at line 2365 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::append()`.

3.11.4.127 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string<_CharT, _Traits, _Alloc>  
std::operator+ ( const _CharT * __lhs, const basic_string<_CharT, _Traits, _Alloc> & __rhs )`

Concatenate C string and string.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

## Returns

New string with value of `__lhs` followed by `__rhs`.

3.11.4.128 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string<_CharT, _Traits, _Alloc>  
std::operator+ ( _CharT __lhs, const basic_string<_CharT, _Traits, _Alloc> & __rhs )`

Concatenate character and string.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

## Returns

New string with `__lhs` followed by `__rhs`.

3.11.4.129 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string<_CharT, _Traits, _Alloc>  
std::operator+ ( const basic_string<_CharT, _Traits, _Alloc> & __lhs, const _CharT * __rhs ) [inline]`

Concatenate string and C string.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

## Returns

New string with `__lhs` followed by `__rhs`.

Definition at line 2402 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::append()`.



3.11.4.130 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string<_CharT, _Traits, _Alloc>  
std::operator+ ( const basic_string<_CharT, _Traits, _Alloc> & __lhs, _CharT __rhs ) [inline]`

Concatenate string and character.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

## Returns

New string with `__lhs` followed by `__rhs`.

Definition at line 2418 of file `basic_string.h`.

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

Stack ordering relation.

## Parameters

<code>__x</code>	A stack.
<code>__y</code>	A stack of the same type as <code>x</code> .

## Returns

True iff `x` is lexicographically less than `__y`.

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with `<`, and `std::lexicographical_compare()` is usually used to make the determination.

Definition at line 261 of file `stl_stack.h`.

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

Queue ordering relation.

## Parameters

<code>__x</code>	A queue.
<code>__y</code>	A queue of the same type as <code>x</code> .

## Returns

True iff `__x` is lexicographically less than `__y`.

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with `<`, and `std::lexicographical_compare()` is usually used to make the determination.

Definition at line 286 of file `stl_queue.h`.

References `std::queue< _Tp, _Sequence>::c`.

**3.11.4.133** `template<typename _Key, typename _Compare, typename _Alloc> bool std::operator< ( const multiset< _Key, _Compare, _Alloc> & __x, const multiset< _Key, _Compare, _Alloc> & __y ) [inline]`

Multiset ordering relation.

**Parameters**

<code>__x</code>	A multiset.
<code>__y</code>	A multiset of the same type as <code>__x</code> .

**Returns**

True iff `__x` is lexicographically less than `__y`.

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 756 of file `stl_multiset.h`.

```
3.11.4.134 template<typename _Key, typename _Compare, typename _Alloc > bool std::operator< ( const set< _Key,
    _Compare, _Alloc > & __x, const set< _Key, _Compare, _Alloc > & __y ) [inline]
```

Set ordering relation.

**Parameters**

<code>__x</code>	A set.
<code>__y</code>	A set of the same type as <code>x</code> .

**Returns**

True iff `__x` is lexicographically less than `__y`.

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 771 of file `stl_set.h`.

```
3.11.4.135 template<typename _Key, typename _Tp, typename _Compare, typename _Alloc > bool std::operator< ( const
    multimap< _Key, _Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp, _Compare, _Alloc > & __y )
    [inline]
```

Multimap ordering relation.

**Parameters**

<code>__x</code>	A multimap.
<code>__y</code>	A multimap of the same type as <code>__x</code> .

**Returns**

True iff `x` is lexicographically less than `y`.

This is a total ordering relation. It is linear in the size of the multimaps. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 881 of file `stl_multimap.h`.

```
3.11.4.136 template<typename _Key, typename _Tp, typename _Compare, typename _Alloc > bool std::operator< ( const
    map< _Key, _Tp, _Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare, _Alloc > & __y ) [inline]
```

Map ordering relation.

## Parameters

<code>__x</code>	A map.
<code>__y</code>	A map of the same type as <code>x</code> .

## Returns

True iff `x` is lexicographically less than `y`.

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 979 of file `stl_map.h`.

```
3.11.4.137 template<typename _Tp, typename _Alloc > bool std::operator< ( const forward_list< _Tp, _Alloc > & __lx, const
forward_list< _Tp, _Alloc > & __ly ) [inline]
```

Forward list ordering relation.

## Parameters

<code>__lx</code>	A forward_list.
<code>__ly</code>	A forward_list of the same type as <code>__lx</code> .

## Returns

True iff `__lx` is lexicographically less than `__ly`.

This is a total ordering relation. It is linear in the number of elements of the forward lists. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1359 of file `forward_list.h`.

References `lexicographical_compare()`.

```
3.11.4.138 template<typename _Tp, typename _Alloc > bool std::operator< ( const vector< _Tp, _Alloc > & __x, const
vector< _Tp, _Alloc > & __y ) [inline]
```

Vector ordering relation.

## Parameters

<code>__x</code>	A vector.
<code>__y</code>	A vector of the same type as <code>__x</code> .

## Returns

True iff `__x` is lexicographically less than `__y`.

This is a total ordering relation. It is linear in the size of the vectors. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1421 of file `stl_vector.h`.

References `lexicographical_compare()`.

3.11.4.139 `template<typename _Tp, typename _Alloc> bool std::operator< ( const list< _Tp, _Alloc> & __x, const list< _Tp, _Alloc> & __y ) [inline]`

List ordering relation.

## Parameters

<code>__x</code>	A list.
<code>__y</code>	A list of the same type as <code>__x</code> .

## Returns

True iff `__x` is lexicographically less than `__y`.

This is a total ordering relation. It is linear in the size of the lists. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1631 of file `stl_list.h`.

References `lexicographical_compare()`.

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

Deque ordering relation.

## Parameters

<code>__x</code>	A deque.
<code>__y</code>	A deque of the same type as <code>__x</code> .

## Returns

True iff `x` is lexicographically less than `__y`.

This is a total ordering relation. It is linear in the size of the deques. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1975 of file `stl_deque.h`.

References `lexicographical_compare()`.

**3.11.4.141** `template<typename _CharT, typename _Traits, typename _Alloc > bool std::operator< ( const basic_string< _CharT, _Traits, _Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs ) [inline]`

Test if string precedes string.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

## Returns

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2569 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

**3.11.4.142** `template<typename _CharT, typename _Traits, typename _Alloc > bool std::operator< ( const basic_string< _CharT, _Traits, _Alloc > & __lhs, const _CharT * __rhs ) [inline]`

Test if string precedes C string.

## Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

## Returns

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2581 of file `basic_string.h`.

3.11.4.143 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator< ( const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs ) [inline]`

Test if C string precedes string.

## Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

## Returns

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2593 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

3.11.4.144 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> basic_ostream< _CharT, _Traits> & std::operator<< ( basic_ostream< _CharT, _Traits > & __os, const __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str ) [inline]`

Write string to a stream.

## Parameters

<code>__os</code>	Output stream.
<code>__str</code>	String to write out.

## Returns

Reference to the output stream.

Output characters of `__str` into `os` following the same rules as for writing a C string.

Definition at line 2521 of file `vstring.h`.

3.11.4.145 `template<typename _CharT, typename _Traits, typename _Alloc> basic_ostream< _CharT, _Traits> & std::operator<< ( basic_ostream< _CharT, _Traits > & __os, const basic_string< _CharT, _Traits, _Alloc > & __str ) [inline]`

Write string to a stream.

## Parameters

<code>__os</code>	Output stream.
<code>__str</code>	String to write out.

## Returns

Reference to the output stream.

Output characters of `__str` into `os` following the same rules as for writing a C string.

Definition at line 2753 of file `basic_string.h`.

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

Based on `operator<`.

Definition at line 279 of file `stl_stack.h`.

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

Based on `operator<`.

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

```
3.11.4.148 template<typename _Key, typename _Compare, typename _Alloc > bool std::operator<= ( const multiset< _Key,
    _Compare, _Alloc > & __x, const multiset< _Key, _Compare, _Alloc > & __y ) [inline]
```

Returns `!(y < x)`

Definition at line 777 of file `stl_multiset.h`.

```
3.11.4.149 template<typename _Key, typename _Compare, typename _Alloc > bool std::operator<= ( const set< _Key,
    _Compare, _Alloc > & __x, const set< _Key, _Compare, _Alloc > & __y ) [inline]
```

Returns `!(y < x)`

Definition at line 792 of file `stl_set.h`.

```
3.11.4.150 template<typename _Key, typename _Tp, typename _Compare, typename _Alloc > bool std::operator<= ( const
    multimap< _Key, _Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp, _Compare, _Alloc > & __y )
    [inline]
```

Based on `operator<`.

Definition at line 902 of file `stl_multimap.h`.

```
3.11.4.151 template<typename _Key, typename _Tp, typename _Compare, typename _Alloc > bool std::operator<= ( const
    map< _Key, _Tp, _Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare, _Alloc > & __y ) [inline]
```

Based on `operator<`.

Definition at line 1000 of file `stl_map.h`.

```
3.11.4.152 template<typename _Tp, typename _Alloc > bool std::operator<= ( const forward_list< _Tp, _Alloc > & __lx,
    const forward_list< _Tp, _Alloc > & __ly ) [inline]
```

Based on `operator<`.



Definition at line 1388 of file forward\_list.h.

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

Based on operator<.

Definition at line 1440 of file stl\_vector.h.

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

Based on operator<.

Definition at line 1650 of file stl\_list.h.

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

Based on operator<.

Definition at line 1997 of file stl\_deque.h.

3.11.4.156 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator<= ( const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs ) [inline]`

Test if string doesn't follow string.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Returns

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2643 of file basic\_string.h.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

3.11.4.157 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator<= ( const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT* __rhs ) [inline]`

Test if string doesn't follow C string.

Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

#### Returns

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2655 of file `basic_string.h`.

**3.11.4.158** `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator<= ( const _CharT* __lhs, const basic_string<_CharT, _Traits, _Alloc> &__rhs ) [inline]`

Test if C string doesn't follow string.

## Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

## Returns

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2667 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

**3.11.4.159** `template<typename _StateT> bool std::operator==( const fpos<_StateT> & __lhs, const fpos<_StateT> & __rhs ) [inline]`

Test if equivalent to another position.

Definition at line 216 of file `postypes.h`.

**3.11.4.160** `template<typename _Tp, typename _Seq> bool std::operator==( const stack<_Tp, _Seq> & __x, const stack<_Tp, _Seq> & __y ) [inline]`

Stack equality comparison.

## Parameters

<code>__x</code>	A stack.
<code>__y</code>	A stack of the same type as <code>__x</code> .

## Returns

True iff the size and elements of the stacks are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and stacks are considered equivalent if their sequences compare equal.

Definition at line 243 of file `stl_stack.h`.

**3.11.4.161** `template<typename _Tp> bool std::operator==( const _Fwd_list_iterator<_Tp> & __x, const _Fwd_list_const_iterator<_Tp> & __y ) [inline]`

Forward list iterator equality comparison.

Definition at line 258 of file `forward_list.h`.

**3.11.4.162** `template<typename _Tp, typename _Seq> bool std::operator==( const queue<_Tp, _Seq> & __x, const queue<_Tp, _Seq> & __y ) [inline]`

Queue equality comparison.

## Parameters

<code>__x</code>	A queue.
------------------	----------

<code>__y</code>	A queue of the same type as <code>__x</code> .
------------------	--

**Returns**

True iff the size and elements of the queues are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and queues are considered equivalent if their sequences compare equal.

Definition at line 268 of file `stl_queue.h`.

References `std::queue<_Tp, _Sequence>::c`.

**3.11.4.163** `template<typename _Key, typename _Compare, typename _Alloc> bool std::operator==( const multiset<_Key, _Compare, _Alloc> &__x, const multiset<_Key, _Compare, _Alloc> &__y ) [inline]`

Multiset equality comparison.

**Parameters**

<code>__x</code>	A multiset.
<code>__y</code>	A multiset of the same type as <code>__x</code> .

**Returns**

True iff the size and elements of the multisets are equal.

This is an equivalence relation. It is linear in the size of the multisets. Multisets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 739 of file `stl_multiset.h`.

**3.11.4.164** `template<typename _Key, typename _Compare, typename _Alloc> bool std::operator==( const set<_Key, _Compare, _Alloc> &__x, const set<_Key, _Compare, _Alloc> &__y ) [inline]`

Set equality comparison.

**Parameters**

<code>__x</code>	A set.
<code>__y</code>	A set of the same type as <code>x</code> .

**Returns**

True iff the size and elements of the sets are equal.

This is an equivalence relation. It is linear in the size of the sets. Sets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 754 of file `stl_set.h`.

**3.11.4.165** `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool std::operator==( const multimap<_Key, _Tp, _Compare, _Alloc> &__x, const multimap<_Key, _Tp, _Compare, _Alloc> &__y ) [inline]`

Multimap equality comparison.

**Parameters**

<code>__x</code>	A multimap.
<code>__y</code>	A multimap of the same type as <code>__x</code> .

**Returns**

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the multimaps. Multimaps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 864 of file `stl_multimap.h`.

```
3.11.4.166 template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator== ( const
map< _Key, _Tp, _Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare, _Alloc > & __y ) [inline]
```

Map equality comparison.

**Parameters**

<code>__x</code>	A map.
<code>__y</code>	A map of the same type as <code>x</code> .

**Returns**

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the maps. Maps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 962 of file `stl_map.h`.

```
3.11.4.167 template<typename _Tp , typename _Alloc > bool std::operator== ( const forward_list< _Tp, _Alloc > & __lx, const
forward_list< _Tp, _Alloc > & __ly )
```

Forward list equality comparison.

**Parameters**

<code>__lx</code>	A <code>forward_list</code>
<code>__ly</code>	A <code>forward_list</code> of the same type as <code>__lx</code> .

**Returns**

True iff the elements of the forward lists are equal.

This is an equivalence relation. It is linear in the number of elements of the forward lists. Deques are considered equivalent if corresponding elements compare equal.

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

Vector equality comparison.

## Parameters

<code>__x</code>	A vector.
<code>__y</code>	A vector of the same type as <code>__x</code> .

## Returns

True iff the size and elements of the vectors are equal.

This is an equivalence relation. It is linear in the size of the vectors. Vectors are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1404 of file `std_vector.h`.

References `std::vector<_Tp, _Alloc>::begin()`, `std::vector<_Tp, _Alloc>::end()`, `equal()`, and `std::vector<_Tp, _Alloc>::size()`.

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

List equality comparison.

## Parameters

<code>__x</code>	A list.
<code>__y</code>	A list of the same type as <code>__x</code> .

## Returns

True iff the size and elements of the lists are equal.

This is an equivalence relation. It is linear in the size of the lists. Lists are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1602 of file `std_list.h`.

References `std::list<_Tp, _Alloc>::begin()`, and `std::list<_Tp, _Alloc>::end()`.

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

Deque equality comparison.

## Parameters

<code>__x</code>	A deque.
<code>__y</code>	A deque of the same type as <code>__x</code> .

## Returns

True iff the size and elements of the deques are equal.

This is an equivalence relation. It is linear in the size of the deques. Deques are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1957 of file `std_deque.h`.

References `std::deque<_Tp, _Alloc>::begin()`, `std::deque<_Tp, _Alloc>::end()`, `equal()`, and `std::deque<_Tp, _Alloc>::size()`.

3.11.4.171 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator==( const basic_string<_CharT, _Traits, _Alloc> & __lhs, const basic_string<_CharT, _Traits, _Alloc> & __rhs ) [inline]`

Test equivalence of two strings.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

## Returns

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 2486 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

**3.11.4.172** `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator==( const _CharT * __lhs, const basic_string<_CharT, _Traits, _Alloc> & __rhs ) [inline]`

Test equivalence of C string and string.

## Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

## Returns

True if `__rhs.compare(__lhs) == 0`. False otherwise.

Definition at line 2507 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

**3.11.4.173** `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator==( const basic_string<_CharT, _Traits, _Alloc> & __lhs, const _CharT * __rhs ) [inline]`

Test equivalence of string and C string.

## Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

## Returns

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 2519 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

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

Based on `operator<`.

Definition at line 273 of file `stl_stack.h`.

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

Based on `operator<`.



Definition at line 298 of file `stl_queue.h`.

3.11.4.176 `template<typename _Key, typename _Compare, typename _Alloc> bool std::operator> ( const multiset< _Key, _Compare, _Alloc> & __x, const multiset< _Key, _Compare, _Alloc> & __y ) [inline]`

Returns  $y < x$ .

Definition at line 770 of file `stl_multiset.h`.

3.11.4.177 `template<typename _Key, typename _Compare, typename _Alloc> bool std::operator> ( const set< _Key, _Compare, _Alloc> & __x, const set< _Key, _Compare, _Alloc> & __y ) [inline]`

Returns  $y < x$ .

Definition at line 785 of file `stl_set.h`.

3.11.4.178 `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool std::operator> ( const multimap< _Key, _Tp, _Compare, _Alloc> & __x, const multimap< _Key, _Tp, _Compare, _Alloc> & __y ) [inline]`

Based on `operator<`.

Definition at line 895 of file `stl_multimap.h`.

3.11.4.179 `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool std::operator> ( const map< _Key, _Tp, _Compare, _Alloc> & __x, const map< _Key, _Tp, _Compare, _Alloc> & __y ) [inline]`

Based on `operator<`.

Definition at line 993 of file `stl_map.h`.

3.11.4.180 `template<typename _Tp, typename _Alloc> bool std::operator> ( const forward_list< _Tp, _Alloc> & __lx, const forward_list< _Tp, _Alloc> & __ly ) [inline]`

Based on `operator<`.

Definition at line 1374 of file `forward_list.h`.

3.11.4.181 `template<typename _Tp, typename _Alloc> bool std::operator> ( const vector< _Tp, _Alloc> & __x, const vector< _Tp, _Alloc> & __y ) [inline]`

Based on `operator<`.

Definition at line 1434 of file `stl_vector.h`.

3.11.4.182 `template<typename _Tp, typename _Alloc> bool std::operator> ( const list< _Tp, _Alloc> & __x, const list< _Tp, _Alloc> & __y ) [inline]`

Based on `operator<`.

Definition at line 1644 of file `stl_list.h`.

3.11.4.183 `template<typename _Tp, typename _Alloc> bool std::operator> ( const deque< _Tp, _Alloc> & __x, const deque< _Tp, _Alloc> & __y ) [inline]`

Based on `operator<`.

Definition at line 1990 of file `stl_deque.h`.

3.11.4.184 `template<typename _CharT, typename _Traits, typename _Alloc > bool std::operator> ( const basic_string<_CharT, _Traits, _Alloc > & __lhs, const basic_string<_CharT, _Traits, _Alloc > & __rhs ) [inline]`

Test if string follows string.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

## Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2606 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

3.11.4.185 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator> ( const basic_string<_CharT, _Traits, _Alloc> & __lhs, const _CharT* __rhs ) [inline]`

Test if string follows C string.

## Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

## Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2618 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

3.11.4.186 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator> ( const _CharT* __lhs, const basic_string<_CharT, _Traits, _Alloc> & __rhs ) [inline]`

Test if C string follows string.

## Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

## Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2630 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

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

Based on operator<.

Definition at line 285 of file `stl_stack.h`.

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

Based on operator<.

Definition at line 310 of file `stl_queue.h`.

```
3.11.4.189 template<typename _Key , typename _Compare , typename _Alloc > bool std::operator>= ( const multiset< _Key,
    _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y ) [inline]
```

Returns `!(x < y)`

Definition at line 784 of file `stl_multiset.h`.

```
3.11.4.190 template<typename _Key , typename _Compare , typename _Alloc > bool std::operator>= ( const set< _Key,
    _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y ) [inline]
```

Returns `!(x < y)`

Definition at line 799 of file `stl_set.h`.

```
3.11.4.191 template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator>= ( const
    multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y )
    [inline]
```

Based on `operator<`.

Definition at line 909 of file `stl_multimap.h`.

```
3.11.4.192 template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator>= ( const
    map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y ) [inline]
```

Based on `operator<`.

Definition at line 1007 of file `stl_map.h`.

```
3.11.4.193 template<typename _Tp , typename _Alloc > bool std::operator>= ( const forward_list< _Tp, _Alloc > &__lx,
    const forward_list< _Tp, _Alloc > &__ly ) [inline]
```

Based on `operator<`.

Definition at line 1381 of file `forward_list.h`.

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

Based on `operator<`.

Definition at line 1446 of file `stl_vector.h`.

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

Based on `operator<`.

Definition at line 1656 of file `stl_list.h`.

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

Based on `operator<`.

Definition at line 2004 of file `stl_deque.h`.

3.11.4.197 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator>= ( const basic_string<_CharT, _Traits, _Alloc> & __lhs, const basic_string<_CharT, _Traits, _Alloc> & __rhs ) [inline]`

Test if string doesn't precede string.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

## Returns

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2680 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

**3.11.4.198** `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator>= ( const basic_string<_CharT, _Traits, _Alloc> & __lhs, const _CharT* __rhs ) [inline]`

Test if string doesn't precede C string.

## Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

## Returns

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2692 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

**3.11.4.199** `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator>= ( const _CharT* __lhs, const basic_string<_CharT, _Traits, _Alloc> & __rhs ) [inline]`

Test if C string doesn't precede string.

## Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

## Returns

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2704 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

**3.11.4.200** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename> class _Base> basic_istream<_CharT, _Traits>& std::operator>> ( basic_istream<_CharT, _Traits> & __is, __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base> & __str )`

Read stream into a string.

**Parameters**

<code>__is</code>	Input stream.
<code>__str</code>	Buffer to store into.

**Returns**

Reference to the input stream.

Stores characters from `__is` into `__str` until whitespace is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased.

**3.11.4.201** `template<typename _CharT, typename _Traits, typename _Alloc > basic_istream<_CharT, _Traits>& std::operator>> ( basic_istream<_CharT, _Traits> & __is, basic_string<_CharT, _Traits, _Alloc> & __str )`

Read stream into a string.

**Parameters**

<code>__is</code>	Input stream.
<code>__str</code>	Buffer to store into.

**Returns**

Reference to the input stream.

Stores characters from `__is` into `__str` until whitespace is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased.

**3.11.4.202** `template<typename _InputIterator, typename _OutputIterator > _OutputIterator std::partial_sum ( _InputIterator __first, _InputIterator __last, _OutputIterator __result )`

Return list of partial sums.

Accumulates the values in the range `[first,last)` using the `+` operator. As each successive input value is added into the total, that partial sum is written to `__result`. Therefore, the first value in `__result` is the first value of the input, the second value in `__result` is the sum of the first and second input values, and so on.

**Parameters**

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Output sum.

**Returns**

Iterator pointing just beyond the values written to `__result`.

Definition at line 237 of file `stl_numeric.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs_pu()`, and `__gnu_parallel::__sequential_random_shuffle()`.

**3.11.4.203** `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation > _OutputIterator std::partial_sum ( _InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op )`

Return list of partial sums.

Accumulates the values in the range `[first,last)` using `__binary_op`. As each successive input value is added into the total, that partial sum is written to `__result`. Therefore, the first value in `__result` is the first value of the input, the second value in `__result` is the sum of the first and second input values, and so on.

#### Parameters

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Output sum.
<code>__binary_op</code>	Function object.

#### Returns

Iterator pointing just beyond the values written to `__result`.

Definition at line 278 of file `stl_numeric.h`.

**3.11.4.204** `template<typename _Inputiterator, typename _Outputiterator, typename _Tp> _Outputiterator std::replace_copy (`  
`_Inputiterator __first, _Inputiterator __last, _Outputiterator __result, const _Tp & __old_value, const _Tp & __new_value`  
`)`

Copy a sequence, replacing each element of one value with another value.

#### Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__old_value</code>	The value to be replaced.
<code>__new_value</code>	The replacement value.

#### Returns

The end of the output sequence, `result+(last-first)`.

Copies each element in the input range `[__first,__last)` to the output range `[__result,__result+(__last-__first))` replacing elements equal to `__old_value` with `__new_value`.

Definition at line 3884 of file `stl_algo.h`.

**3.11.4.205** `template<typename _Tp> void std::return_temporary_buffer ( _Tp * __p ) [inline]`

The companion to `get_temporary_buffer()`.

#### Parameters

<code>__p</code>	A buffer previously allocated by <code>get_temporary_buffer</code> .
------------------	--

#### Returns

None.

Frees the memory pointed to by `__p`.

Definition at line 112 of file `stl_tempbuf.h`.

Referenced by `std::_Temporary_buffer<_ForwardIterator, _Tp>::_Temporary_buffer()`.



**3.11.4.206 ios\_base& std::right ( ios\_base & \_\_base ) [inline]**

Calls base.setf(ios\_base::right, ios\_base::adjustfield).

Definition at line 924 of file ios\_base.h.

References \_\_gnu\_debug::\_\_base(), std::ios\_base::adjustfield, std::ios\_base::right, and std::ios\_base::setf().

**3.11.4.207 ios\_base& std::scientific ( ios\_base & \_\_base ) [inline]**

Calls base.setf(ios\_base::scientific, ios\_base::floatfield).

Definition at line 966 of file ios\_base.h.

References \_\_gnu\_debug::\_\_base(), std::ios\_base::floatfield, std::ios\_base::scientific, and std::ios\_base::setf().

**3.11.4.208 ios\_base& std::showbase ( ios\_base & \_\_base ) [inline]**

Calls base.setf(ios\_base::showbase).

Definition at line 811 of file ios\_base.h.

References \_\_gnu\_debug::\_\_base(), std::ios\_base::setf(), and std::ios\_base::showbase.

**3.11.4.209 ios\_base& std::showpoint ( ios\_base & \_\_base ) [inline]**

Calls base.setf(ios\_base::showpoint).

Definition at line 827 of file ios\_base.h.

References \_\_gnu\_debug::\_\_base(), std::ios\_base::setf(), and std::ios\_base::showpoint.

**3.11.4.210 ios\_base& std::showpos ( ios\_base & \_\_base ) [inline]**

Calls base.setf(ios\_base::showpos).

Definition at line 843 of file ios\_base.h.

References \_\_gnu\_debug::\_\_base(), std::ios\_base::setf(), and std::ios\_base::showpos.

**3.11.4.211 ios\_base& std::skipws ( ios\_base & \_\_base ) [inline]**

Calls base.setf(ios\_base::skipws).

Definition at line 859 of file ios\_base.h.

References \_\_gnu\_debug::\_\_base(), std::ios\_base::setf(), and std::ios\_base::skipws.

**3.11.4.212 template<typename \_Tp, typename \_Tp1, \_Lock\_policy \_Lp> \_\_shared\_ptr<\_Tp, \_Lp> std::static\_pointer\_cast ( const \_\_shared\_ptr<\_Tp1, \_Lp> & \_\_r ) [inline], [noexcept]**

static\_pointer\_cast

Definition at line 1157 of file shared\_ptr\_base.h.

**3.11.4.213 template<typename \_Key, typename \_Compare, typename \_Alloc> void std::swap ( multiset<\_Key, \_Compare, \_Alloc> & \_\_x, multiset<\_Key, \_Compare, \_Alloc> & \_\_y ) [inline]**

See std::multiset::swap().

Definition at line 791 of file stl\_multiset.h.

References std::multiset<\_Key, \_Compare, \_Alloc>::swap().

3.11.4.214 `template<typename _Key, typename _Compare, typename _Alloc> void std::swap ( set< _Key, _Compare, _Alloc> & __x, set< _Key, _Compare, _Alloc> & __y ) [inline]`

See `std::set::swap()`.

Definition at line 806 of file `stl_set.h`.

References `std::set< _Key, _Compare, _Alloc>::swap()`.

3.11.4.215 `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> void std::swap ( multimap< _Key, _Tp, _Compare, _Alloc> & __x, multimap< _Key, _Tp, _Compare, _Alloc> & __y ) [inline]`

See `std::multimap::swap()`.

Definition at line 916 of file `stl_multimap.h`.

References `std::multimap< _Key, _Tp, _Compare, _Alloc>::swap()`.

3.11.4.216 `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> void std::swap ( map< _Key, _Tp, _Compare, _Alloc> & __x, map< _Key, _Tp, _Compare, _Alloc> & __y ) [inline]`

See `std::map::swap()`.

Definition at line 1014 of file `stl_map.h`.

References `std::map< _Key, _Tp, _Compare, _Alloc>::swap()`.

3.11.4.217 `template<typename _Tp, typename _Alloc> void std::swap ( forward_list< _Tp, _Alloc> & __lx, forward_list< _Tp, _Alloc> & __ly ) [inline]`

See `std::forward_list::swap()`.

Definition at line 1395 of file `forward_list.h`.

References `std::forward_list< _Tp, _Alloc>::swap()`.

3.11.4.218 `template<typename _Tp, typename _Alloc> void std::swap ( vector< _Tp, _Alloc> & __x, vector< _Tp, _Alloc> & __y ) [inline]`

See `std::vector::swap()`.

Definition at line 1452 of file `stl_vector.h`.

References `std::vector< _Tp, _Alloc>::swap()`.

3.11.4.219 `template<typename _Tp, typename _Alloc> void std::swap ( list< _Tp, _Alloc> & __x, list< _Tp, _Alloc> & __y ) [inline]`

See `std::list::swap()`.

Definition at line 1662 of file `stl_list.h`.

References `std::list< _Tp, _Alloc>::swap()`.

3.11.4.220 `template<typename _Tp, typename _Alloc> void std::swap ( deque< _Tp, _Alloc> & __x, deque< _Tp, _Alloc> & __y ) [inline]`

See `std::deque::swap()`.

Definition at line 2011 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc>::swap()`.

3.11.4.221 `template<typename _CharT, typename _Traits, typename _Alloc> void std::swap ( basic_string<_CharT, _Traits, _Alloc> &__lhs, basic_string<_CharT, _Traits, _Alloc> &__rhs ) [inline]`

Swap contents of two strings.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Exchanges the contents of `__lhs` and `__rhs` in constant time.

Definition at line 2717 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::swap()`.

**3.11.4.222** `template<typename _CharT> _CharT std::tolower ( _CharT __c, const locale & __loc ) [inline]`

Convenience interface to `ctype.tolower(__c)`.

Definition at line 2602 of file `locale_facets.h`.

**3.11.4.223** `template<typename _CharT> _CharT std::toupper ( _CharT __c, const locale & __loc ) [inline]`

Convenience interface to `ctype.toupper(__c)`.

Definition at line 2596 of file `locale_facets.h`.

**3.11.4.224** `template<typename _InputIterator, typename _ForwardIterator> _ForwardIterator std::uninitialized_copy ( _InputIterator __first, _InputIterator __last, _ForwardIterator __result ) [inline]`

Copies the range `[first,last)` into `result`.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.

## Returns

`__result + (__last - __first)`

Like `copy()`, but does not require an initialized output range.

Definition at line 107 of file `stl_uninitialized.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms_pu()`.

**3.11.4.225** `template<typename _InputIterator, typename _Size, typename _ForwardIterator> _ForwardIterator std::uninitialized_copy_n ( _InputIterator __first, _Size __n, _ForwardIterator __result ) [inline]`

Copies the range `[first,first+n)` into `result`.

## Parameters

<code>__first</code>	An input iterator.
<code>__n</code>	The number of elements to copy.
<code>__result</code>	An output iterator.

## Returns

`__result + __n`

Like `copy_n()`, but does not require an initialized output range.

Definition at line 647 of file `stl_uninitialized.h`.

References `__iterator_category()`.

**3.11.4.226** `template<typename _ForwardIterator, typename _Tp> void std::uninitialized_fill ( _ForwardIterator __first, _ForwardIterator __last, const _Tp & __x ) [inline]`

Copies the value `x` into the range `[first,last)`.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__x</code>	The source value.

Returns

Nothing.

Like `fill()`, but does not require an initialized output range.

Definition at line 164 of file `stl_uninitialized.h`.

**3.11.4.227** `template<typename _ForwardIterator, typename _Size, typename _Tp> void std::uninitialized_fill_n ( _ForwardIterator __first, _Size __n, const _Tp & __x ) [inline]`

Copies the value `x` into the range `[first,first+n)`.

Parameters

<code>__first</code>	An input iterator.
<code>__n</code>	The number of copies to make.
<code>__x</code>	The source value.

Returns

Nothing.

Like `fill_n()`, but does not require an initialized output range.

Definition at line 218 of file `stl_uninitialized.h`.

**3.11.4.228** `ios_base& std::unitbuf ( ios_base & __base ) [inline]`

Calls `base.setf(ios_base::unitbuf)`.

Definition at line 891 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::setf()`, and `std::ios_base::unitbuf`.

**3.11.4.229** `ios_base& std::uppercase ( ios_base & __base ) [inline]`

Calls `base.setf(ios_base::uppercase)`.

Definition at line 875 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::setf()`, and `std::ios_base::uppercase`.

## 3.12 `std::__debug` Namespace Reference

- class `map`
- class `multimap`
- class `multiset`
- class `set`

[illegible]

- `template<typename _Key, typename _Compare, typename _Allocator> bool operator>= (const set< _Key, _Compare, _Allocator> &__lhs, const set< _Key, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> bool operator>= (const multimap< _Key, _Tp, _Compare, _Allocator> &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> bool operator>= (const map< _Key, _Tp, _Compare, _Allocator> &__lhs, const map< _Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator> void swap (multiset< _Key, _Compare, _Allocator> &__x, multiset< _Key, _Compare, _Allocator> &__y)`
- `template<typename _Key, typename _Compare, typename _Allocator> void swap (set< _Key, _Compare, _Allocator> &__x, set< _Key, _Compare, _Allocator> &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> void swap (multimap< _Key, _Tp, _Compare, _Allocator> &__lhs, multimap< _Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> void swap (map< _Key, _Tp, _Compare, _Allocator> &__lhs, map< _Key, _Tp, _Compare, _Allocator> &__rhs)`

### 3.12.1 Detailed Description

GNU debug code, replaces standard behavior with debug behavior.

Macros and namespaces used by the implementation outside of debug wrappers to verify certain properties. The `__glibcxx_requires_xxx` macros are merely wrappers around the `__glibcxx_check_xxx` wrappers when we are compiling with debug mode, but disappear when we are in release mode so that there is no checking performed in, e.g., the standard library algorithms.

## 3.13 `std::` `__detail` Namespace Reference

### Classes

- class [\\_Automaton](#)
- struct [\\_Before\\_begin](#)
- struct [\\_CharMatcher](#)
- class [\\_Compiler](#)
- struct [\\_Default\\_ranged\\_hash](#)
- struct [\\_EndTagger](#)
- struct [\\_Equal\\_helper](#)
- struct [\\_Equal\\_helper< \\_Key, \\_Value, \\_ExtractKey, \\_Equal, \\_HashCodeType, false >](#)
- struct [\\_Equal\\_helper< \\_Key, \\_Value, \\_ExtractKey, \\_Equal, \\_HashCodeType, true >](#)
- struct [\\_Equality](#)
- struct [\\_Equality< \\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits, false >](#)
- struct [\\_Equality< \\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits, true >](#)
- struct [\\_Equality\\_base](#)
- class [\\_Grep\\_matcher](#)
- struct [\\_Hash\\_code\\_base](#)
- struct [\\_Hash\\_code\\_base< \\_Key, \\_Value, \\_ExtractKey, \\_H1, \\_H2, \\_Default\\_ranged\\_hash, false >](#)
- struct [\\_Hash\\_code\\_base< \\_Key, \\_Value, \\_ExtractKey, \\_H1, \\_H2, \\_Default\\_ranged\\_hash, true >](#)
- struct [\\_Hash\\_code\\_base< \\_Key, \\_Value, \\_ExtractKey, \\_H1, \\_H2, \\_Hash, false >](#)
- struct [\\_Hash\\_node](#)
- struct [\\_Hash\\_node< \\_Value, false >](#)
- struct [\\_Hash\\_node< \\_Value, true >](#)
- struct [\\_Hash\\_node\\_base](#)
- struct [\\_Hashtable\\_base](#)

- struct [\\_Hashtable\\_ebo\\_helper](#)
- struct [\\_Hashtable\\_ebo\\_helper< \\_Nm, \\_Tp, false >](#)
- struct [\\_Hashtable\\_ebo\\_helper< \\_Nm, \\_Tp, true >](#)
- struct [\\_Hashtable\\_traits](#)
- struct [\\_Insert](#)
- struct [\\_Insert< \\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits, false, \\_↵ Unique\\_keys >](#)
- struct [\\_Insert< \\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits, true, false >](#)
- struct [\\_Insert< \\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits, true, true >](#)
- struct [\\_Insert\\_base](#)
- struct [\\_List\\_node\\_base](#)
- struct [\\_Local\\_const\\_iterator](#)
- struct [\\_Local\\_iterator](#)
- struct [\\_Local\\_iterator\\_base](#)
- struct [\\_Local\\_iterator\\_base< \\_Key, \\_Value, \\_ExtractKey, \\_H1, \\_H2, \\_Hash, false >](#)
- struct [\\_Local\\_iterator\\_base< \\_Key, \\_Value, \\_ExtractKey, \\_H1, \\_H2, \\_Hash, true >](#)
- struct [\\_Map\\_base](#)
- struct [\\_Map\\_base< \\_Key, \\_Pair, \\_Alloc, \\_Select1st, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits, false >](#)
- struct [\\_Map\\_base< \\_Key, \\_Pair, \\_Alloc, \\_Select1st, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits, true >](#)
- struct [\\_Mod\\_range\\_hashing](#)
- class [\\_Nfa](#)
- struct [\\_Node\\_const\\_iterator](#)
- struct [\\_Node\\_iterator](#)
- struct [\\_Node\\_iterator\\_base](#)
- struct [\\_PatternCursor](#)
- struct [\\_Prime\\_rehash\\_policy](#)
- struct [\\_RangeMatcher](#)
- struct [\\_Rehash\\_base](#)
- struct [\\_Rehash\\_base< \\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_Prime\\_rehash\\_policy, \\_Traits >](#)
- struct [\\_Results](#)
- class [\\_Scanner](#)
- struct [\\_Scanner\\_base](#)
- class [\\_SpecializedCursor](#)
- class [\\_SpecializedResults](#)
- struct [\\_StartTagger](#)
- struct [\\_State](#)
- class [\\_StateSeq](#)

#### Typedefs

- typedef [std::shared\\_ptr< \\_Automaton > \\_AutomatonPtr](#)
- typedef [std::function< bool\(const \\_PatternCursor &\)> \\_Matcher](#)
- typedef [int \\_StateIdT](#)
- typedef [std::set< \\_StateIdT > \\_StateSet](#)
- typedef [std::stack< \\_StateIdT, std::vector< \\_StateIdT > > \\_StateStack](#)
- typedef [std::function< void\(const \\_PatternCursor &, \\_Results &\)> \\_Tagger](#)



## Enumerations

- enum [\\_Opcode](#) { [\\_S\\_opcode\\_unknown](#), [\\_S\\_opcode\\_alternative](#), [\\_S\\_opcode\\_subexpr\\_begin](#), [\\_S\\_opcode\\_subexpr\\_end](#), [\\_S\\_opcode\\_match](#), [\\_S\\_opcode\\_accept](#) }

## Functions

- template<typename \_InIter, typename \_TraitsT > [\\_AutomatonPtr](#) **\_\_compile** (const \_InIter &\_\_b, const \_InIter &\_\_e, \_TraitsT &\_\_t, [regex\\_constants::syntax\\_option\\_type](#) \_\_f)
- template<typename \_FwdIterT > [\\_SpecializedCursor](#)< \_FwdIterT > **\_\_cursor** (const \_FwdIterT &\_\_b, const \_FwdIterT &\_\_e)
- template<class \_Iterator > std::iterator\_traits< \_Iterator >::difference\_type **\_\_distance\_fw** (\_Iterator \_\_first, \_Iterator \_\_last, [std::input\\_iterator\\_tag](#))
- template<class \_Iterator > std::iterator\_traits< \_Iterator >::difference\_type **\_\_distance\_fw** (\_Iterator \_\_first, \_Iterator \_\_last, [std::forward\\_iterator\\_tag](#))
- template<class \_Iterator > std::iterator\_traits< \_Iterator >::difference\_type **\_\_distance\_fw** (\_Iterator \_\_first, \_Iterator \_\_last)
- bool [\\_AnyMatcher](#) (const [\\_PatternCursor](#) &)
- template<typename \_Value, bool \_Cache\_hash\_code> bool **operator!=** (const [\\_Node\\_iterator\\_base](#)< \_Value, \_Cache\_hash\_code > &\_\_x, const [\\_Node\\_iterator\\_base](#)< \_Value, \_Cache\_hash\_code > &\_\_y)
- template<typename \_Key, typename \_Value, typename \_ExtractKey, typename \_H1, typename \_H2, typename \_Hash, bool \_\_cache> bool **operator!=** (const [\\_Local\\_iterator\\_base](#)< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_cache > &\_\_x, const [\\_Local\\_iterator\\_base](#)< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_cache > &\_\_y)
- template<typename \_Value, bool \_Cache\_hash\_code> bool **operator==** (const [\\_Node\\_iterator\\_base](#)< \_Value, \_Cache\_hash\_code > &\_\_x, const [\\_Node\\_iterator\\_base](#)< \_Value, \_Cache\_hash\_code > &\_\_y)
- template<typename \_Key, typename \_Value, typename \_ExtractKey, typename \_H1, typename \_H2, typename \_Hash, bool \_\_cache> bool **operator==** (const [\\_Local\\_iterator\\_base](#)< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_cache > &\_\_x, const [\\_Local\\_iterator\\_base](#)< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_cache > &\_\_y)

## Variables

- static const [\\_StateIdT](#) [\\_S\\_invalid\\_state\\_id](#)

## 3.13.1 Detailed Description

Implementation details not part of the namespace std interface.

## 3.14 std::\_\_parallel Namespace Reference

## Classes

- struct [\\_CRandNumber](#)

## Functions

- template<typename \_Iter, typename \_Tp, typename \_Tag > \_Tp **\_\_accumulate\_switch** (\_Iter, \_Iter, \_Tp, \_Tag)
- template<typename \_Iter, typename \_Tp, typename \_BinaryOper, typename \_Tag > \_Tp **\_\_accumulate\_switch** (\_Iter, \_Iter, \_Tp, \_BinaryOper, \_Tag)

- `template<typename _RAIter, typename _Tp, typename _BinaryOper> _Tp __accumulate_switch (_RAIter, _RAIter, _Tp, _BinaryOper, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism=__gnu_parallel::__parallel_unbalanced)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2> _OIter __adjacent_difference_switch (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper> _OIter __adjacent_difference_switch (_Iter, _Iter, _OIter, _BinaryOper, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism=__gnu_parallel::__parallel_unbalanced)`
- `template<typename _Filter, typename _IterTag> _Filter __adjacent_find_switch (_Filter, _Filter, _IterTag)`
- `template<typename _Filter, typename _BiPredicate, typename _IterTag> _Filter __adjacent_find_switch (_Filter, _Filter, _BiPredicate, _IterTag)`
- `template<typename _RAIter, typename _BiPredicate> _RAIter __adjacent_find_switch (_RAIter, _RAIter, _BiPredicate, random_access_iterator_tag)`
- `template<typename _RAIter> _RAIter __adjacent_find_switch (_RAIter __begin, _RAIter __end, random_access_iterator_tag)`
- `template<typename _FIterator, typename _IteratorTag> _FIterator __adjacent_find_switch (_FIterator __begin, _FIterator __end, _IteratorTag)`
- `template<typename _FIterator, typename _BinaryPredicate, typename _IteratorTag> _FIterator __adjacent_find_switch (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _BinaryPredicate> _RAIter __adjacent_find_switch (_RAIter __begin, _RAIter __end, _BinaryPredicate __pred, random_access_iterator_tag)`
- `template<typename _Iter, typename _Predicate, typename _IterTag> iterator_traits< _Iter >::difference_type __count_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate> iterator_traits< _RAIter >::difference_type __count_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu_parallel::__parallel_unbalanced)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag> iterator_traits< _Iter >::difference_type __count_if_switch (_Iter __begin, _Iter __end, _Predicate __pred, _IteratorTag)`
- `template<typename _Iter, typename _Tp, typename _IterTag> iterator_traits< _Iter >::difference_type __count_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp> iterator_traits< _RAIter >::difference_type __count_switch (_RAIter __begin, _RAIter __end, const _Tp & __value, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu_parallel::__parallel_unbalanced)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag> iterator_traits< _Iter >::difference_type __count_switch (_Iter __begin, _Iter __end, const _Tp & __value, _IteratorTag)`
- `template<typename _Iter, typename _Filter, typename _IterTag1, typename _IterTag2> _Iter __find_first_of_switch (_Iter, _Iter, _Filter, _Filter, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _Filter, typename _BiPredicate, typename _IterTag> _RAIter __find_first_of_switch (_RAIter, _RAIter, _Filter, _Filter, _BiPredicate, random_access_iterator_tag, _IterTag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate, typename _IterTag1, typename _IterTag2> _Iter __find_first_of_switch (_Iter, _Iter, _Filter, _Filter, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _Iter, typename _FIterator, typename _IteratorTag1, typename _IteratorTag2> _Iter __find_first_of_switch (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag> _RAIter __find_first_of_switch (_RAIter __begin1, _RAIter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, random_access_iterator_tag, _IteratorTag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2> _Iter __find_first_of_switch (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _Predicate, typename _IterTag> _Iter __find_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag> _Iter __find_if_switch (_Iter __begin, _Iter __end, _Predicate __pred, _IteratorTag)`

- `template<typename _RAIter, typename _Predicate> _RAIter __find_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag> _Iter __find_switch (_Iter __begin, _Iter __end, const _Tp & __val, _IteratorTag)`
- `template<typename _RAIter, typename _Tp> _RAIter __find_switch (_RAIter __begin, _RAIter __end, const _Tp & __val, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Tp, typename _IterTag> _Iter __find_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _Iter, typename _Function, typename _IteratorTag> _Function __for_each_switch (_Iter __begin, _Iter __end, _Function __f, _IteratorTag)`
- `template<typename _RAIter, typename _Function> _Function __for_each_switch (_RAIter __begin, _RAIter __end, _Function __f, random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _Iter, typename _Function, typename _IterTag> _Function __for_each_switch (_Iter, _Iter, _Function, _IterTag)`
- `template<typename _OIter, typename _Size, typename _Generator, typename _IterTag> _OIter __generate_n_switch (_OIter, _Size, _Generator, _IterTag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator, typename _IteratorTag> _OutputIterator __generate_n_switch (_OutputIterator __begin, _Size __n, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter, typename _Size, typename _Generator> _RAIter __generate_n_switch (_RAIter __begin, _Size __n, _Generator __gen, random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _Filter, typename _Generator, typename _IterTag> void __generate_switch (_Filter, _Filter, _Generator, _IterTag)`
- `template<typename _Filterator, typename _Generator, typename _IteratorTag> void __generate_switch (_Filterator __begin, _Filterator __end, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter, typename _Generator> void __generate_switch (_RAIter __begin, _RAIter __end, _Generator __gen, random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename BinaryFunction1, typename BinaryFunction2> _Tp __inner_product_switch (_RAIter1, _RAIter1, _RAIter2, _Tp, BinaryFunction1, BinaryFunction2, random\_access\_iterator\_tag, random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism=\_\_gnu\_parallel::parallel\_unbalanced)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _Tag1, typename _Tag2> _Tp __inner_product_switch (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, _Tag1, _Tag2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2> bool __lexicographical_compare_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate> bool __lexicographical_compare_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2> bool __lexicographical_compare_switch (_Iter1, _Iter1, _Iter2, _Iter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _Filter, typename _Compare, typename _IterTag> _Filter __max_element_switch (_Filter, _Filter, _Compare, _IterTag)`
- `template<typename _Filterator, typename _Compare, typename _IteratorTag> _Filterator __max_element_switch (_Filterator __begin, _Filterator __end, _Compare __comp, _IteratorTag)`
- `template<typename _RAIter, typename _Compare> _RAIter __max_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare, typename _IterTag1, typename _IterTag2, typename _IterTag3> _OIter __merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, _IterTag1, _IterTag2, _IterTag3)`

- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare > _OIter __merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 > _OutputIterator __merge_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare > _OutputIterator __merge_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Filter, typename _Compare, typename _IterTag > _Filter __min_element_switch (_Filter, _Filter, _Compare, _IterTag)`
- `template<typename _Filterator, typename _Compare, typename _IteratorTag > _Filterator __min_element_switch (_Filterator __begin, _Filterator __end, _Compare __comp, _IteratorTag)`
- `template<typename _RAIter, typename _Compare > _RAIter __min_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag=gnu\_parallel::parallel\_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 > pair< _Iter1, _Iter2 > __mismatch_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate > pair< _RAIter1, _RAIter2 > __mismatch_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2 > pair< _Iter1, _Iter2 > __mismatch_switch (_Iter1, _Iter1, _Iter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2 > _OIter __partial_sum_switch (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper > _OIter __partial_sum_switch (_Iter, _Iter, _OIter, _BinaryOper, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Filter, typename _Predicate, typename _IterTag > _Filter __partition_switch (_Filter, _Filter, _Predicate, _IterTag)`
- `template<typename _Filterator, typename _Predicate, typename _IteratorTag > _Filterator __partition_switch (_Filterator __begin, _Filterator __end, _Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate > _RAIter __partition_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random\_access\_iterator\_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp, typename _IterTag > void __replace_if_switch (_Filter, _Filter, _Predicate, const _Tp &, _IterTag)`
- `template<typename _Filterator, typename _Predicate, typename _Tp, typename _IteratorTag > void __replace_if_switch (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp & __new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate, typename _Tp > void __replace_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp & __new_value, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag=gnu\_parallel::parallel\_balanced)`
- `template<typename _Filter, typename _Tp, typename _IterTag > void __replace_switch (_Filter, _Filter, const _Tp &, const _Tp &, _IterTag)`
- `template<typename _Filterator, typename _Tp, typename _IteratorTag > void __replace_switch (_Filterator __begin, _Filterator __end, const _Tp & __old_value, const _Tp & __new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Tp > void __replace_switch (_RAIter __begin, _RAIter __end, const _Tp & __old_value, const _Tp & __new_value, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag=gnu\_parallel::parallel\_balanced)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BiPredicate > _RAIter __search_n_switch (_RAIter, _RAIter, _Integer, const _Tp &, _BiPredicate, random\_access\_iterator\_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate, typename _IterTag > _Filter __search_n_switch (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, _IterTag)`

- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BinaryPredicate > _RAIter __search_n_switch (↵  
_RAIter __begin, _RAIter __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, random\_↵  
\_\_access\_iterator\_tag)`
- `template<typename _Filterator, typename _Integer, typename _Tp, typename _BinaryPredicate, typename _IteratorTag > _Filterator  
__search_n_switch (_Filterator __begin, _Filterator __end, _Integer __count, const _Tp &__val, _Binary↵  
Predicate __binary_pred, _IteratorTag)`
- `template<typename _Filter1, typename _Filter2, typename _IterTag1, typename _IterTag2 > _Filter1 __search_switch (_Filter1,  
_Filter1, _Filter2, _Filter2, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BiPredicate > _RAIter1 __search_switch (_RAIter1, _RAIter1,  
_RAIter2, _RAIter2, _BiPredicate, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate, typename _IterTag1, typename _IterTag2 > _Filter1 __↵  
search_switch (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2 > _RAIter1 __search_switch (_RAIter1 __begin1, _RAIter1 __end1,  
_RAIter2 __begin2, _RAIter2 __end2, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Filterator1, typename _Filterator2, typename _IteratorTag1, typename _IteratorTag2 > _Filterator1 __search↵  
__switch (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2, _IteratorTag1,  
_IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BinaryPredicate > _RAIter1 __search_switch (_RAIter1 __↵  
begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _BinaryPredicate __pred, random\_access\_↵  
iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Filterator1, typename _Filterator2, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2  
> _Filterator1 __search_switch (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2  
__end2, _BinaryPredicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename ↵  
_IteratorTag2, typename _IteratorTag3 > _OutputIterator __set_difference_switch (_Iter1 __begin1, _Iter1 __end1,  
_Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, ↵  
_IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate > _Output_RAIter __set↵  
difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RA↵  
Iter __result, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_↵  
iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, type-  
name _IterTag3 > _OIter __set_difference_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1,  
_IterTag2, _IterTag3)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename ↵  
_IteratorTag2, typename _IteratorTag3 > _OutputIterator __set_intersection_switch (_Iter1 __begin1, _Iter1 __end1,  
_Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, ↵  
_IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate > _Output_RAIter __set↵  
intersection_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_R↵  
Alter __result, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_↵  
iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, type-  
name _IterTag3 > _OIter __set_intersection_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1,  
_IterTag2, _IterTag3)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename ↵  
_IteratorTag2, typename _IteratorTag3 > _OutputIterator __set_symmetric_difference_switch (_Iter1 __begin1, _Iter1  
__end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _Iterator↵  
Tag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate > _Output_RAIter __↵  
set_symmetric_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __↵  
end2, _Output_RAIter __result, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag,  
random\_access\_iterator\_tag)`

- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3 > _OIter __set_symmetric_difference_switch ( _Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 > _OutputIterator __set_union_switch ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate > _Output_RAIter __set_union_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3 > _OIter __set_union_switch ( _Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation, typename _IterTag1, typename _IterTag2 > _OIter __transform1_switch ( _Iter, _Iter, _OIter, _UnaryOperation, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RAOIter, typename _UnaryOperation > _RAOIter __transform1_switch ( _RAIter, _RAIter, _RAOIter, _UnaryOperation, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism=__gnu_parallel::parallel_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation > _RAIter2 __transform1_switch ( _RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation, typename _IteratorTag1, typename _IteratorTag2 > _RAIter2 __transform1_switch ( _RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BiOperation > _RAIter3 __transform2_switch ( _RAIter1, _RAIter1, _RAIter2, _RAIter3, _BiOperation, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism=__gnu_parallel::parallel_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation, typename _Tag1, typename _Tag2, typename _Tag3 > _OIter __transform2_switch ( _Iter1, _Iter1, _Iter2, _OIter, _BiOperation, _Tag1, _Tag2, _Tag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BinaryOperation > _RAIter3 __transform2_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter3 __result, _BinaryOperation __binary_op, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation, typename _Tag1, typename _Tag2, typename _Tag3 > _OutputIterator __transform2_switch ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, _Tag1, _Tag2, _Tag3)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 > _OutputIterator __unique_copy_switch ( _Iter __begin, _Iter __last, _OutputIterator __out, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename RandomAccessOutputIterator, typename _Predicate > RandomAccessOutputIterator __unique_copy_switch ( _RAIter __begin, _RAIter __last, RandomAccessOutputIterator __out, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _IterTag1, typename _IterTag2 > _OIter __unique_copy_switch ( _Iter, _Iter, _OIter, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RandomAccess_OIter, typename _Predicate > _RandomAccess_OIter __unique_copy_switch ( _RAIter, _RAIter, _RandomAccess_OIter, _Predicate, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp > _Tp accumulate ( _Iter, _Iter, _Tp)`
- `template<typename _Iter, typename _Tp > _Tp accumulate ( _Iter, _Iter, _Tp, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp > _Tp accumulate ( _Iter, _Iter, _Tp, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper > _Tp accumulate ( _Iter, _Iter, _Tp, _BinaryOper)`



- `template<typename _Iter, typename _Tp, typename _BinaryOper > _Tp accumulate (_Iter, _Iter, _Tp, _BinaryOper, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper > _Tp accumulate (_Iter, _Iter, _Tp, _BinaryOper, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter > _OIter adjacent_difference (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper > _OIter adjacent_difference (_Iter, _Iter, _OIter, __BinaryOper, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter > _OIter adjacent_difference (_Iter, _Iter, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper > _OIter adjacent_difference (_Iter, _Iter, _OIter, __BinaryOper, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter > _OIter adjacent_difference (_Iter, _Iter, _OIter, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper > _OIter adjacent_difference (_Iter, _Iter, _OIter, __BinaryOper, __gnu_parallel::Parallelism)`
- `template<typename _Filter > _Filter adjacent_find (_Filter, _Filter)`
- `template<typename _Filter > _Filter adjacent_find (_Filter, _Filter, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _BiPredicate > _Filter adjacent_find (_Filter, _Filter, _BiPredicate)`
- `template<typename _Filter, typename _BiPredicate > _Filter adjacent_find (_Filter, _Filter, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator > _FIterator adjacent_find (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _BinaryPredicate > _FIterator adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator > _FIterator adjacent_find (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _BinaryPredicate > _FIterator adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred)`
- `template<typename _Iter, typename _Tp > iterator_traits< _Iter >::difference_type count (_Iter __begin, _Iter __end, const _Tp &__value, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp > iterator_traits< _Iter >::difference_type count (_Iter __begin, _Iter __end, const _Tp &__value, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp > iterator_traits< _Iter >::difference_type count (_Iter __begin, _Iter __end, const _Tp &__value)`
- `template<typename _Iter, typename _Predicate > iterator_traits< _Iter >::difference_type count_if (_Iter __begin, _Iter __end, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Predicate > iterator_traits< _Iter >::difference_type count_if (_Iter __begin, _Iter __end, _Predicate __pred, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate > iterator_traits< _Iter >::difference_type count_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2 > bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate > bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 > bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate > bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _Iter, typename _Tp > _Iter find (_Iter __begin, _Iter __end, const _Tp &__val, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp > _Iter find (_Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _Filter > _Iter find_first_of (_Iter, _Iter, _Filter, _Filter, __gnu_parallel::sequential_tag)`

- `template<typename _Iter, typename _Filter, typename _BiPredicate> _Iter find_first_of (_Iter, _Iter, _Filter, _Filter, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate> _Iter find_first_of (_Iter, _Iter, _Filter, _Filter, _BiPredicate)`
- `template<typename _Iter, typename _Filter> _Iter find_first_of (_Iter, _Iter, _Filter, _Filter)`
- `template<typename _Iter, typename _FIterator> _Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate> _Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate> _Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp)`
- `template<typename _Iter, typename _FIterator> _Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2)`
- `template<typename _Iter, typename _Predicate> _Iter find_if (_Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate> _Iter find_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Function> _Function for_each (_Iter __begin, _Iter __end, _Function __f, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iterator, typename _Function> _Function for_each (_Iterator __begin, _Iterator __end, _Function __f, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iterator, typename _Function> _Function for_each (_Iterator __begin, _Iterator __end, _Function __f)`
- `template<typename _Iter, typename _Function> _Function for_each (_Iter, _Iter, _Function)`
- `template<typename _Filter, typename _Generator> void generate (_Filter, _Filter, _Generator)`
- `template<typename _Filter, typename _Generator> void generate (_Filter, _Filter, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Generator> void generate (_Filter, _Filter, _Generator, \_\_gnu\_parallel::Parallelism)`
- `template<typename _FIterator, typename _Generator> void generate (_FIterator __begin, _FIterator __end, _Generator __gen, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Generator> void generate (_FIterator __begin, _FIterator __end, _Generator __gen, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Generator> void generate (_FIterator __begin, _FIterator __end, _Generator __gen)`
- `template<typename _OIter, typename _Size, typename _Generator> _OIter generate_n (_OIter, _Size, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator> _OIter generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OIter, typename _Size, typename _Generator> _OIter generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::Parallelism)`
- `template<typename _OutputIterator, typename _Size, typename _Generator> _OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator> _OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator> _OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen)`
- `template<typename _Iter1, typename _Iter2, typename _Tp> _Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp)`
- `template<typename _Iter1, typename _Iter2, typename _Tp> _Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp> _Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2> _Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2)`



- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 > _Tp inner←  
_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename BinaryFunction1, typename BinaryFunction2 > _Tp inner←  
_product (_Iter1, _Iter1, _Iter2, _Tp, BinaryFunction1, BinaryFunction2, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2 > bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2  
__begin2, _Iter2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate > bool lexicographical_compare (_Iter1 __begin1,  
_Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 > bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2  
__begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate > bool lexicographical_compare (_Iter1 __begin1,  
_Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred)`
- `template<typename _Filter > _Filter max_element (_Filter, _Filter)`
- `template<typename _Filter > _Filter max_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter > _Filter max_element (_Filter, _Filter, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Compare > _Filter max_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare > _Filter max_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::←  
::sequential\_tag)`
- `template<typename _Filter, typename _Compare > _Filter max_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::←  
Parallelism)`
- `template<typename _Filterator > _Filterator max_element (_Filterator __begin, _Filterator __end, \_\_gnu\_parallel::←  
::sequential\_tag)`
- `template<typename _Filterator, typename _Compare > _Filterator max_element (_Filterator __begin, _Filterator __end,  
_Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator > _Filterator max_element (_Filterator __begin, _Filterator __end, \_\_gnu\_parallel::←  
Parallelism __parallelism_tag)`
- `template<typename _Filterator > _Filterator max_element (_Filterator __begin, _Filterator __end)`
- `template<typename _Filterator, typename _Compare > _Filterator max_element (_Filterator __begin, _Filterator __end,  
_Compare __comp, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filterator, typename _Compare > _Filterator max_element (_Filterator __begin, _Filterator __end,  
_Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter > _OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter,  
\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare > _OIter merge (_Iter1, _Iter1, _Iter2,  
_Iter2, _OIter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare > _OIter merge (_Iter1, _Iter1, _Iter2,  
_Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter > _OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator > _OutputIterator merge (_Iter1 __begin1, _Iter1  
__end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare > _OutputIterator merge (_I←  
Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp,  
\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare > _OutputIterator merge (_Iter1  
__begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator > _OutputIterator merge (_Iter1 __begin1, _Iter1  
__end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result)`
- `template<typename _Filter > _Filter min_element (_Filter, _Filter)`
- `template<typename _Filter > _Filter min_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter > _Filter min_element (_Filter, _Filter, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Compare > _Filter min_element (_Filter, _Filter, _Compare)`

- `template<typename _Filter, typename _Compare> _Filter min_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Compare> _Filter min_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _FIterator> _FIterator min_element (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Compare> _FIterator min_element (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::sequential\_tag, \_\_gnu\_parallel::Parallelism)`
- `template<typename _FIterator> _FIterator min_element (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::Parallelism, \_\_gnu\_parallel::parallelism\_tag)`
- `template<typename _FIterator> _FIterator min_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _Compare> _FIterator min_element (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::Parallelism, \_\_gnu\_parallel::parallelism\_tag, \_\_gnu\_parallel::Compare)`
- `template<typename _FIterator, typename _Compare> _FIterator min_element (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::Compare)`
- `template<typename _Iter1, typename _Iter2> pair<_Iter1, _Iter2> mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate> pair<_Iter1, _Iter2> mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2> pair<_Iter1, _Iter2> mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate> pair<_Iter1, _Iter2> mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _RAIter> void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare> void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, \_\_gnu\_parallel::sequential\_tag, \_\_gnu\_parallel::Compare)`
- `template<typename _RAIter, typename _Compare> void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, \_\_gnu\_parallel::Compare)`
- `template<typename _RAIter> void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end)`
- `template<typename _RAIter, typename _Compare> void partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, \_\_gnu\_parallel::sequential\_tag, \_\_gnu\_parallel::Compare)`
- `template<typename _RAIter> void partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare> void partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, \_\_gnu\_parallel::Compare)`
- `template<typename _RAIter> void partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end)`
- `template<typename _Iter, typename _OIter> _OIter partial_sum (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper> _OIter partial_sum (_Iter, _Iter, _OIter, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter> _OIter partial_sum (_Iter, _Iter, _OIter, \_\_gnu\_parallel::result)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper> _OIter partial_sum (_Iter, _Iter, _OIter, _BinaryOper)`
- `template<typename _Filter, typename _Predicate> _Filter partition (_Filter, _Filter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Predicate> _Filter partition (_Filter, _Filter, _Predicate)`
- `template<typename _FIterator, typename _Predicate> _FIterator partition (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::sequential\_tag, \_\_gnu\_parallel::Predicate)`
- `template<typename _FIterator, typename _Predicate> _FIterator partition (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::Predicate)`
- `template<typename _RAIter> void random_shuffle (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _RAIter, typename _RandomNumberGenerator> void random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rand, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter> void random_shuffle (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter, typename _RandomNumberGenerator> void random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &&__rand)`
- `template<typename _Filter, typename _Tp> void replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _Filter, typename _Tp> void replace (_Filter, _Filter, const _Tp &, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Tp> void replace (_Filter, _Filter, const _Tp &, const _Tp &, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filterator, typename _Tp> void replace (_Filterator __begin, _Filterator __end, const _Tp &__old_value, const _Tp &__new_value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator, typename _Tp> void replace (_Filterator __begin, _Filterator __end, const _Tp &__old_value, const _Tp &__new_value, \_\_gnu\_parallel::Parallelism parallelism_tag)`
- `template<typename _Filterator, typename _Tp> void replace (_Filterator __begin, _Filterator __end, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _Filter, typename _Predicate, typename _Tp> void replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _Filter, typename _Predicate, typename _Tp> void replace_if (_Filter, _Filter, _Predicate, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp> void replace_if (_Filter, _Filter, _Predicate, const _Tp &, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filterator, typename _Predicate, typename _Tp> void replace_if (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp &__new_value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator, typename _Predicate, typename _Tp> void replace_if (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp &__new_value, \_\_gnu\_parallel::Parallelism parallelism_tag)`
- `template<typename _Filterator, typename _Predicate, typename _Tp> void replace_if (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _Filter1, typename _Filter2> _Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter1, typename _Filter2> _Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate> _Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate> _Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate)`
- `template<typename _Filterator1, typename _Filterator2> _Filterator1 search (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator1, typename _Filterator2> _Filterator1 search (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2)`
- `template<typename _Filterator1, typename _Filterator2, typename _BinaryPredicate> _Filterator1 search (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2, _BinaryPredicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator1, typename _Filterator2, typename _BinaryPredicate> _Filterator1 search (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2, _BinaryPredicate __pred)`
- `template<typename _Filter, typename _Integer, typename _Tp> _Filter search_n (_Filter, _Filter, _Integer, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate> _Filter search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp> _Filter search_n (_Filter, _Filter, _Integer, const _Tp &)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate> _Filter search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate)`
- `template<typename _Filterator, typename _Integer, typename _Tp> _Filterator search_n (_Filterator __begin, _Filterator __end, _Integer __count, const _Tp &__val, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate> _FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp> _FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate> _FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator> _OutputIterator set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate> _OutputIterator set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator> _OutputIterator set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate> _OutputIterator set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _IIter1, typename _IIter2, typename _OIter> _OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate> _OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter> _OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate> _OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator> _OutputIterator set_intersection (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate> _OutputIterator set_intersection (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator> _OutputIterator set_intersection (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate> _OutputIterator set_intersection (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _IIter1, typename _IIter2, typename _OIter> _OIter set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate> _OIter set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter> _OIter set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate> _OIter set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator> _OutputIterator set_symmetric_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate> _OutputIterator set_symmetric_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator> _OutputIterator set_symmetric_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate> _OutputIterator set_symmetric_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred)`

- `template<typename _Iter1, typename _Iter2, typename _OIter> _OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate> _OIter set_symmetric_difference (↵ _Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter> _OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate> _OIter set_symmetric_difference (↵ _Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator> _OutputIterator set_union (_Iter1 __begin1, ↵ _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate> _OutputIterator set_union (↵ _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator> _OutputIterator set_union (_Iter1 __begin1, ↵ _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate> _OutputIterator set_union (↵ _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OIter> _OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate> _OIter set_union (_Iter1, _Iter1, ↵ _Iter2, _Iter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter> _OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate> _OIter set_union (_Iter1, _Iter1, ↵ _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _RAIter> void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare> void sort (_RAIter __begin, _RAIter __end, _Compare __comp, ↵ \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism> void sort (_RAIter __begin, _RAIter __end, ↵ _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter> void sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter> void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::default\_parallel\_tag ↵ __parallelism)`
- `template<typename _RAIter> void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::parallel\_tag __parallelism)`
- `template<typename _RAIter> void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_tag __parallelism)`
- `template<typename _RAIter> void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort ↵ sampling\_tag __parallelism)`
- `template<typename _RAIter> void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort ↵ exact\_tag __parallelism)`
- `template<typename _RAIter> void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::quicksort\_tag ↵ __parallelism)`
- `template<typename _RAIter> void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::balanced\_quicksort\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare> void sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter> void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare> void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, ↵ \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism> void stable_sort (_RAIter __begin, _RAIter ↵ __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter> void stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter> void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::default\_parallel\_tag __parallelism)`

- `template<typename _RAIter > void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::parallel\_tag __↵  
parallelism)`
- `template<typename _RAIter > void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_↵  
mergesort\_tag __parallelism)`
- `template<typename _RAIter > void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::quicksort\_tag __↵  
parallelism)`
- `template<typename _RAIter > void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::balanced\_↵  
quicksort\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare > void stable_sort (_RAIter __begin, _RAIter __end, _Compare __↵  
comp)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation > _OIter transform (_Iter, _Iter, _OIter, _Unary↵  
Operation)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation > _OIter transform (_Iter, _Iter, _OIter, _Unary↵  
Operation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation > _OIter transform (_Iter, _Iter, _OIter, _Unary↵  
Operation, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation > _OIter transform (_Iter1, _Iter1,↵  
_Iter2, _OIter, _BiOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation > _OIter transform (_Iter1, _Iter1,↵  
_Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation > _OIter transform (_Iter1, _Iter1,↵  
_Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation > _OutputIterator transform (_Iter __begin,↵  
_Iter __end, _OutputIterator __result, _UnaryOperation __unary_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation > _OutputIterator transform (_Iter __↵  
begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op, \_\_gnu\_parallel::Parallelism __↵  
parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation > _OutputIterator transform (_Iter __begin,↵  
_Iter __end, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation > _OutputIterator transform↵  
(_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, ↵  
\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation > _OutputIterator transform↵  
(_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, ↵  
\_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation > _OutputIterator transform↵  
(_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _OutputIterator > _OutputIterator unique_copy (_Iter __begin1, _Iter __end1, ↵  
_OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate > _OutputIterator unique_copy (_Iter __begin1,↵  
_Iter __end1, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator > _OutputIterator unique_copy (_Iter __begin1, _Iter __end1, ↵  
_OutputIterator __out)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate > _OutputIterator unique_copy (_Iter __begin1,↵  
_Iter __end1, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter, typename _OIter > _OIter unique_copy (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _Predicate > _OIter unique_copy (_Iter, _Iter, _OIter, _Predicate,↵  
\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter > _OIter unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _Predicate > _OIter unique_copy (_Iter, _Iter, _OIter, _Predicate)`



### 3.14.1 Detailed Description

GNU parallel code, replaces standard behavior with parallel behavior.

## 3.15 `std::__profile` Namespace Reference

### Classes

- class [map](#)
- class [multimap](#)
- class [multiset](#)
- class [set](#)

### Functions

- `template<typename _UnorderedCont, typename _Value, bool _Cache_hash_code> bool __are_equal (const _UnorderedCont &__uc, const \_\_detail::\_\_Hash\_node< _Value, _Cache_hash_code > *__lhs, const \_\_detail::\_\_Hash\_node< _Value, _Cache_hash_code > *__rhs)`
- `template<typename _UnorderedCont, typename _Value, bool _Cache_hash_code> std::size_t __get_bucket_index (const _UnorderedCont &__uc, const \_\_detail::\_\_Hash\_node< _Value, _Cache_hash_code > *__node)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence> bool operator!= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence> bool operator!= (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator> bool operator!= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> bool operator!= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator> bool operator!= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> bool operator!= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Iterator, typename _Sequence> __iterator_tracker< _Iterator, _Sequence > operator+ (typename __iterator_tracker< _Iterator, _Sequence >::difference_type __n, const __iterator_tracker< _Iterator, _Sequence > &__i)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence> __iterator_tracker< _IteratorL, _Sequence >::difference_type operator- (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence> __iterator_tracker< _Iterator, _Sequence >::difference_type operator- (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence> bool operator< (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence> bool operator< (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator> bool operator< (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> bool operator< (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator> bool operator< (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`

- Generated on Tue Jun 30 2015 12:19:40 for libstdc++ by Doxygen



- `template<typename _Key , typename _Compare , typename _Allocator > void swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y)`
- `template<typename _Key , typename _Tp , typename _Compare , typename _Allocator > void swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Tp , typename _Compare , typename _Allocator > void swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare, _Allocator > &__rhs)`

### 3.15.1 Detailed Description

GNU profile code, replaces standard behavior with profile behavior.

## 3.16 std::regex\_constants Namespace Reference

### 5.1 Regular Expression Syntax Options

- `enum __syntax_option { _S_icode, _S_nosubs, _S_optimize, _S_collate, _S_ECMAScript, _S_basic, _S_↵ extended, _S_awk, _S_grep, _S_egrep, _S_syntax_last }`
- `typedef unsigned int syntax_option_type`
- `constexpr syntax_option_type icode`
- `constexpr syntax_option_type nosubs`
- `constexpr syntax_option_type optimize`
- `constexpr syntax_option_type collate`
- `constexpr syntax_option_type ECMAScript`
- `constexpr syntax_option_type basic`
- `constexpr syntax_option_type extended`
- `constexpr syntax_option_type awk`
- `constexpr syntax_option_type grep`
- `constexpr syntax_option_type egrep`

### 5.2 Matching Rules

Matching a regular expression against a sequence of characters [first, last) proceeds according to the rules of the grammar specified for the regular expression object, modified according to the effects listed below for any bitmask elements set.

- `enum __match_flag { _S_not_bol, _S_not_eol, _S_not_bow, _S_not_eow, _S_any, _S_not_null, _S_↵ continuous, _S_prev_avail, _S_sed, _S_no_copy, _S_first_only, _S_match_flag_last }`
- `typedef std::bitset< _S_match_flag_last > match_flag_type`
- `constexpr match_flag_type match_default`
- `constexpr match_flag_type match_not_bol`
- `constexpr match_flag_type match_not_eol`
- `constexpr match_flag_type match_not_bow`
- `constexpr match_flag_type match_not_eow`
- `constexpr match_flag_type match_any`
- `constexpr match_flag_type match_not_null`
- `constexpr match_flag_type match_continuous`
- `constexpr match_flag_type match_prev_avail`
- `constexpr match_flag_type format_default`
- `constexpr match_flag_type format_sed`
- `constexpr match_flag_type format_no_copy`
- `constexpr match_flag_type format_first_only`

## 5.3 Error Types

- enum `error_type` { `_S_error_collate`, `_S_error_ctype`, `_S_error_escape`, `_S_error_backref`, `_S_error_↵brack`, `_S_error_paren`, `_S_error_brace`, `_S_error_badbrace`, `_S_error_range`, `_S_error_space`, `_S_error_↵_badrepeat`, `_S_error_complexity`, `_S_error_stack`, `_S_error_last` }
- constexpr `error_type error_collate` (`_S_error_collate`)
- constexpr `error_type error_ctype` (`_S_error_ctype`)
- constexpr `error_type error_escape` (`_S_error_escape`)
- constexpr `error_type error_backref` (`_S_error_backref`)
- constexpr `error_type error_brack` (`_S_error_brack`)
- constexpr `error_type error_paren` (`_S_error_paren`)
- constexpr `error_type error_brace` (`_S_error_brace`)
- constexpr `error_type error_badbrace` (`_S_error_badbrace`)
- constexpr `error_type error_range` (`_S_error_range`)
- constexpr `error_type error_space` (`_S_error_space`)
- constexpr `error_type error_badrepeat` (`_S_error_badrepeat`)
- constexpr `error_type error_complexity` (`_S_error_complexity`)
- constexpr `error_type error_stack` (`_S_error_stack`)

## 3.16.1 Detailed Description

ISO C++-0x entities sub namespace for regex.

## 3.17 std::rel\_ops Namespace Reference

## Functions

- template<class `_Tp` > bool `operator!=` (`const _Tp &__x`, `const _Tp &__y`)
- template<class `_Tp` > bool `operator<=` (`const _Tp &__x`, `const _Tp &__y`)
- template<class `_Tp` > bool `operator>` (`const _Tp &__x`, `const _Tp &__y`)
- template<class `_Tp` > bool `operator>=` (`const _Tp &__x`, `const _Tp &__y`)

## 3.17.1 Detailed Description

The generated relational operators are sequestered here.

## 3.17.2 Function Documentation

3.17.2.1 template<class `_Tp` > bool std::rel\_ops::operator!= ( `const _Tp &__x`, `const _Tp &__y` ) [inline]

Defines != for arbitrary types, in terms of ==.

## Parameters

<code>__x</code>	A thing.
<code>__y</code>	Another thing.

**Returns**

`__x != __y`

This function uses `==` to determine its result.

Definition at line 87 of file `stl_relops.h`.

**3.17.2.2** `template<class _Tp> bool std::rel_ops::operator<= ( const _Tp & __x, const _Tp & __y ) [inline]`

Defines `<=` for arbitrary types, in terms of `<`.

**Parameters**

<code>__x</code>	A thing.
<code>__y</code>	Another thing.

**Returns**

`__x <= __y`

This function uses `<` to determine its result.

Definition at line 113 of file `stl_relops.h`.

**3.17.2.3** `template<class _Tp> bool std::rel_ops::operator> ( const _Tp & __x, const _Tp & __y ) [inline]`

Defines `>` for arbitrary types, in terms of `<`.

**Parameters**

<code>__x</code>	A thing.
<code>__y</code>	Another thing.

**Returns**

`__x > __y`

This function uses `<` to determine its result.

Definition at line 100 of file `stl_relops.h`.

**3.17.2.4** `template<class _Tp> bool std::rel_ops::operator>= ( const _Tp & __x, const _Tp & __y ) [inline]`

Defines `>=` for arbitrary types, in terms of `<`.

**Parameters**

<code>__x</code>	A thing.
<code>__y</code>	Another thing.

**Returns**

`__x >= __y`

This function uses `<` to determine its result.

Definition at line 126 of file `stl_relops.h`.

### 3.18 std::tr1 Namespace Reference

#### Namespaces

- [\\_\\_detail](#)

#### 3.18.1 Detailed Description

ISO C++ TR1 entities toplevel namespace is std::tr1.

### 3.19 std::tr1::\_\_detail Namespace Reference

#### 3.19.1 Detailed Description

Implementation details not part of the namespace std::tr1 interface.

### 3.20 std::tr2 Namespace Reference

#### Namespaces

- [\\_\\_detail](#)

#### 3.20.1 Detailed Description

ISO C++ TR2 entities toplevel namespace is std::tr2.

### 3.21 std::tr2::\_\_detail Namespace Reference

#### 3.21.1 Detailed Description

Implementation details not part of the namespace std::tr2 interface.

## 4 Class Documentation

### 4.1 \_\_cxxabiv1::\_\_forced\_unwind Class Reference

#### 4.1.1 Detailed Description

Thrown as part of forced unwinding.

A magic placeholder class that can be caught by reference to recognize forced unwinding.

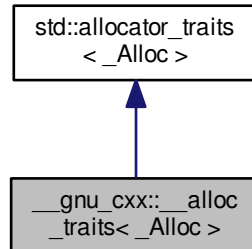
Definition at line 48 of file cxxabi\_forced.h.

The documentation for this class was generated from the following file:

- [cxxabi\\_forced.h](#)

## 4.2 `__gnu_cxx::__alloc_traits<_Alloc>` Struct Template Reference

Inheritance diagram for `__gnu_cxx::__alloc_traits<_Alloc>`:



### Public Types

- typedef `std::allocator_traits<_Alloc>` **\_Base\_type**
- typedef `_Alloc` **allocator\_type**
- typedef `_Base_type::const_pointer` **const\_pointer**
- typedef `const value_type & const_reference`
- typedef `__const_void_pointer` **const\_void\_pointer**
- typedef `_Base_type::difference_type` **difference\_type**
- typedef `_Base_type::pointer` **pointer**
- typedef `__propagate_on_container_copy_assignment` **propagate\_on\_container\_copy\_assignment**
- typedef `__propagate_on_container_move_assignment` **propagate\_on\_container\_move\_assignment**
- typedef `__propagate_on_container_swap` **propagate\_on\_container\_swap**
- template<typename `_Tp`> using **rebind\_alloc** = `typename __alloc_traits::rebind_alloc<_Alloc, _Tp>::__type`
- template<typename `_Tp`> using **rebind\_traits** = `allocator_traits<rebind_alloc<_Tp>>`
- typedef `value_type & reference`
- typedef `_Base_type::size_type` **size\_type**
- typedef `_Base_type::value_type` **value\_type**
- typedef `__void_pointer` **void\_pointer**

### Static Public Member Functions

- static constexpr bool **\_S\_always\_equal** ()
- static constexpr bool **\_S\_nothrow\_move** ()
- static constexpr bool **\_S\_nothrow\_swap** ()
- static void **\_S\_on\_swap** (`_Alloc &__a, _Alloc &__b`)
- static constexpr bool **\_S\_propagate\_on\_copy\_assign** ()
- static constexpr bool **\_S\_propagate\_on\_move\_assign** ()
- static constexpr bool **\_S\_propagate\_on\_swap** ()
- static `_Alloc` **\_S\_select\_on\_copy** (`const _Alloc &__a`)
- static `pointer allocate` (`_Alloc &__a, size_type __n`)

- static `pointer allocate` (`_Alloc &__a`, `size_type __n`, `const_void_pointer __hint`)
- `template<typename _Ptr, typename... _Args> static std::enable_if<__is_custom_pointer<_Ptr>::value>::type construct` (`_Alloc &__a`, `_Ptr __p`, `_Args &&...__args`)
- `template<typename _Tp, typename... _Args> static auto construct` (`_Alloc &__a`, `_Tp *__p`, `_Args &&...__args`) -> `decltype(_S_construct(__a, __p, std::forward<_Args>(__args)...))`
- static void `deallocate` (`_Alloc &__a`, `pointer __p`, `size_type __n`)
- `template<typename _Ptr> static std::enable_if<__is_custom_pointer<_Ptr>::value>::type destroy` (`_Alloc &__a`, `_Ptr __p`)
- `template<class _Tp> static void destroy` (`_Alloc &__a`, `_Tp *__p`)
- static `size_type max_size` (`const _Alloc &__a`)
- static `_Alloc select_on_container_copy_construction` (`const _Alloc &__rhs`)

#### 4.2.1 Detailed Description

`template<typename _Alloc> struct __gnu_cxx::__alloc_traits<_Alloc>`

Uniform interface to C++98 and C++0x allocators.

Definition at line 121 of file `ext/alloc_traits.h`.

#### 4.2.2 Member Typedef Documentation

4.2.2.1 `template<typename _Alloc> typedef __const_void_pointer std::allocator_traits<_Alloc>::const_void_pointer`  
[`inherited`]

The allocator's const void pointer type.

`Alloc::const_void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const void>`

Definition at line 140 of file `bits/alloc_traits.h`.

4.2.2.2 `template<typename _Alloc> typedef __propagate_on_container_copy_assignment std::allocator_traits<_Alloc>::propagate_on_container_copy_assignment` [`inherited`]

How the allocator is propagated on copy assignment.

`Alloc::propagate_on_container_copy_assignment` if that type exists, otherwise `false_type`

Definition at line 174 of file `bits/alloc_traits.h`.

4.2.2.3 `template<typename _Alloc> typedef __propagate_on_container_move_assignment std::allocator_traits<_Alloc>::propagate_on_container_move_assignment` [`inherited`]

How the allocator is propagated on move assignment.

`Alloc::propagate_on_container_move_assignment` if that type exists, otherwise `false_type`

Definition at line 186 of file `bits/alloc_traits.h`.

4.2.2.4 `template<typename _Alloc> typedef __propagate_on_container_swap std::allocator_traits<_Alloc>::propagate_on_container_swap` [`inherited`]

How the allocator is propagated on swap.

`Alloc::propagate_on_container_swap` if that type exists, otherwise `false_type`

Definition at line 197 of file `bits/alloc_traits.h`.

**4.2.2.5** `template<typename _Alloc> typedef __void_pointer std::allocator_traits< _Alloc >::void_pointer`  
`[inherited]`

The allocator's void pointer type.

`Alloc::void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<void>`

Definition at line 129 of file `bits/alloc_traits.h`.

### 4.2.3 Member Function Documentation

**4.2.3.1** `template<typename _Alloc> static pointer std::allocator_traits< _Alloc >::allocate ( _Alloc & __a, size_type __n )`  
`[inline], [static], [inherited]`

Allocate memory.

Parameters

<code>__a</code>	An allocator.
<code>__n</code>	The number of objects to allocate space for.

Calls `a.allocate(n)`

Definition at line 350 of file `bits/alloc_traits.h`.

**4.2.3.2** `template<typename _Alloc> static pointer std::allocator_traits< _Alloc >::allocate ( _Alloc & __a, size_type __n,`  
`const_void_pointer __hint ) [inline], [static], [inherited]`

Allocate memory.

Parameters

<code>__a</code>	An allocator.
<code>__n</code>	The number of objects to allocate space for.
<code>__hint</code>	Aid to locality.

Returns

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

Definition at line 365 of file `bits/alloc_traits.h`.

**4.2.3.3** `template<typename _Alloc> template<typename _Tp, typename... _Args> static auto std::allocator_traits<`  
`_Alloc >::construct ( _Alloc & __a, _Tp * __p, _Args &&... __args ) -> decltype( _S_construct( __a, __p,`  
`std::forward< _Args>( __args )... ) ) [inline], [static], [inherited]`

Construct an object of type `_Tp`.

Parameters

<code>__a</code>	An allocator.
<code>__p</code>	Pointer to memory of suitable size and alignment for <code>Tp</code>
<code>__args</code>	Constructor arguments.

Calls `__a.construct( __p, std::forward<Args>( __args )... )` if that expression is well-formed, otherwise uses placement-new to construct an object of type `_Tp` at location `__p` from the arguments `__args...`

Definition at line 391 of file `bits/alloc_traits.h`.

4.2.3.4 `template<typename _Alloc> static void std::allocator_traits<_Alloc>::deallocate ( _Alloc & __a, pointer __p, size_type __n )` `[inline]`, `[static]`, `[inherited]`

Deallocate memory.



## Parameters

<code>__a</code>	An allocator.
<code>__p</code>	Pointer to the memory to deallocate.
<code>__n</code>	The number of objects space was allocated for.

Calls `a.deallocate(p, n)`

Definition at line 376 of file `bits/alloc_traits.h`.

**4.2.3.5** `template<typename _Alloc> template<class _Tp> static void std::allocator_traits<_Alloc>::destroy ( _Alloc & __a, _Tp* __p ) [inline], [static], [inherited]`

Destroy an object of type `_Tp`.

## Parameters

<code>__a</code>	An allocator.
<code>__p</code>	Pointer to the object to destroy

Calls `__a.destroy(__p)` if that expression is well-formed, otherwise calls `__p->~_Tp()`

Definition at line 404 of file `bits/alloc_traits.h`.

**4.2.3.6** `template<typename _Alloc> static size_type std::allocator_traits<_Alloc>::max_size ( const _Alloc & __a ) [inline], [static], [inherited]`

The maximum supported allocation size.

## Parameters

<code>__a</code>	An allocator.
------------------	---------------

## Returns

`__a.max_size()` or `numeric_limits<size_type>::max()`

Returns `__a.max_size()` if that expression is well-formed, otherwise returns `numeric_limits<size_type>::max()`

Definition at line 415 of file `bits/alloc_traits.h`.

Referenced by `std::forward_list<_Tp, _Alloc>::max_size()`.

**4.2.3.7** `template<typename _Alloc> static _Alloc std::allocator_traits<_Alloc>::select_on_container_copy_construction ( const _Alloc & __rhs ) [inline], [static], [inherited]`

Obtain an allocator to use when copying a container.

## Parameters

<code>__rhs</code>	An allocator.
--------------------	---------------

## Returns

`__rhs.select_on_container_copy_construction()` or `__rhs`

Returns `__rhs.select_on_container_copy_construction()` if that expression is well-formed, otherwise returns `__rhs`

Definition at line 427 of file `bits/alloc_traits.h`.

The documentation for this struct was generated from the following file:

- [ext/alloc\\_traits.h](#)

### 4.3 `__gnu_cxx::__common_pool_policy<_PoolTp, _Thread >` Struct Template Reference

Inherits `__gnu_cxx::__common_pool_base<_PoolTp, _Thread >`.

#### 4.3.1 Detailed Description

```
template<template< bool > class _PoolTp, bool _Thread>struct __gnu_cxx::__common_pool_policy< _PoolTp, _Thread >
```

Policy for shared `__pool` objects.

Definition at line 460 of file `mt_allocator.h`.

The documentation for this struct was generated from the following file:

- [mt\\_allocator.h](#)

### 4.4 `__gnu_cxx::__detail::__mini_vector<_Tp >` Class Template Reference

#### Public Types

- typedef const `_Tp` & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef pointer **iterator**
- typedef `_Tp *` **pointer**
- typedef `_Tp &` **reference**
- typedef size\_t **size\_type**
- typedef `_Tp` **value\_type**

#### Public Member Functions

- reference **back** () const throw ()
- iterator **begin** () const throw ()
- void **clear** () throw ()
- iterator **end** () const throw ()
- void **erase** (iterator \_\_pos) throw ()
- void **insert** (iterator \_\_pos, const\_reference \_\_x)
- reference **operator[]** (const size\_type \_\_pos) const throw ()
- void **pop\_back** () throw ()
- void **push\_back** (const\_reference \_\_x)
- size\_type **size** () const throw ()

#### 4.4.1 Detailed Description

```
template<typename _Tp>class __gnu_cxx::__detail::__mini_vector<_Tp >
```

`__mini_vector<>` is a stripped down version of the full-fledged `std::vector<>`.

It is to be used only for built-in types or PODs. Notable differences are:

1. Not all accessor functions are present. 2. Used ONLY for PODs. 3. No Allocator template argument. Uses operator new() to get memory, and operator delete() to free it. Caveat: The dtor does NOT free the memory allocated, so this a memory-leaking vector!

Definition at line 69 of file `bitmap_allocator.h`.

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

## 4.5 `__gnu_cxx::__detail::_Bitmap_counter<_Tp>` Class Template Reference

### Public Member Functions

- `_Bitmap_counter` (`_BPVector` &Rvbp, long \_\_index=-1)
- pointer `_M_base` () const throw ()
- bool `_M_finished` () const throw ()
- `size_t * _M_get` () const throw ()
- `_Index_type _M_offset` () const throw ()
- void `_M_reset` (long \_\_index=-1) throw ()
- void `_M_set_internal_bitmap` (`size_t * __new_internal_marker`) throw ()
- `_Index_type _M_where` () const throw ()
- `_Bitmap_counter` & `operator++` () throw ()

### 4.5.1 Detailed Description

```
template<typename _Tp>class __gnu_cxx::__detail::_Bitmap_counter<_Tp>
```

The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.

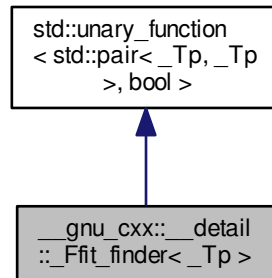
Definition at line 396 of file `bitmap_allocator.h`.

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

4.6 `__gnu_cxx::__detail::_Ffit_finder<_Tp>` Class Template Reference

Inheritance diagram for `__gnu_cxx::__detail::_Ffit_finder<_Tp>`:



## Public Types

- typedef `std::pair<_Tp, _Tp>` `argument_type`
- typedef `bool` `result_type`

## Public Member Functions

- `size_t * _M_get () const throw ()`
- `_Counter_type _M_offset () const throw ()`
- `bool operator() (_Block_pair __bp) throw ()`

## 4.6.1 Detailed Description

```
template<typename _Tp>class __gnu_cxx::__detail::_Ffit_finder<_Tp>
```

The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.

Definition at line 331 of file `bitmap_allocator.h`.

## 4.6.2 Member Typedef Documentation

4.6.2.1 `typedef std::pair<_Tp, _Tp> std::unary_function< std::pair<_Tp, _Tp>, bool>::argument_type`  
`[inherited]`

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

#### 4.6.2.2 `typedef bool std::unary_function< std::pair<_Tp, _Tp>, bool>::result_type` [inherited]

`result_type` is the return type

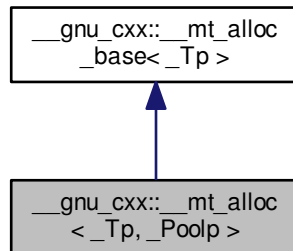
Definition at line 107 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

### 4.7 `__gnu_cxx::__mt_alloc<_Tp, _Poolp>` Class Template Reference

Inheritance diagram for `__gnu_cxx::__mt_alloc<_Tp, _Poolp>`:



#### Public Types

- `typedef _Poolp __policy_type`
- `typedef _Poolp::pool_type __pool_type`
- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef std::true_type propagate_on_container_move_assignment`
- `typedef _Tp & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

#### Public Member Functions

- `__mt_alloc` (const `__mt_alloc` &) noexcept
- `template<typename _Tp1, typename _Poolp1> __mt_alloc` (const `__mt_alloc<_Tp1, _Poolp1>` &) noexcept
- `const __pool_base::_Tune M_get_options` ()
- `void M_set_options` (`__pool_base::_Tune` \_\_t)
- `pointer address` (reference \_\_x) const noexcept
- `const_pointer address` (const\_reference \_\_x) const noexcept

- pointer **allocate** (size\_type \_\_n, const void \*=0)
- template<typename \_Up, typename... \_Args> void **construct** (\_Up \*\_\_p, \_Args &&...\_\_args)
- void **deallocate** (pointer \_\_p, size\_type \_\_n)
- template<typename \_Up> void **destroy** (\_Up \*\_\_p)
- size\_type **max\_size** () const noexcept

#### 4.7.1 Detailed Description

```
template<typename _Tp, typename _Poolp = __common_pool_policy<__pool, true>>class __gnu_cxx::__mt_alloc<_Tp, _Poolp>
```

This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a *global* one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the *global* list).

Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html>.

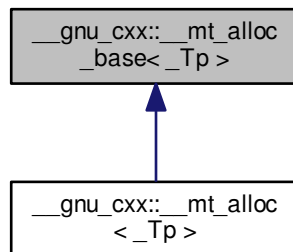
Definition at line 639 of file `mt_allocator.h`.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

## 4.8 `__gnu_cxx::__mt_alloc_base<_Tp>` Class Template Reference

Inheritance diagram for `__gnu_cxx::__mt_alloc_base<_Tp>`:



#### Public Types

- typedef const \_Tp \* **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef \_Tp \* **pointer**
- typedef std::true\_type **propagate\_on\_container\_move\_assignment**
- typedef \_Tp & **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- pointer **address** (reference \_\_x) const noexcept
- const\_pointer **address** (const\_reference \_\_x) const noexcept
- template<typename \_Up, typename... \_Args> void **construct** (\_Up \*\_\_p, \_Args &&... \_\_args)
- template<typename \_Up> void **destroy** (\_Up \*\_\_p)
- size\_type **max\_size** () const noexcept

### 4.8.1 Detailed Description

```
template<typename _Tp>class __gnu_cxx::__mt_alloc_base< _Tp >
```

Base class for \_Tp dependent member functions.

Definition at line 570 of file mt\_allocator.h.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

## 4.9 \_\_gnu\_cxx::\_\_per\_type\_pool\_policy< \_Tp, \_PoolTp, \_Thread > Struct Template Reference

Inherits \_\_gnu\_cxx::\_\_per\_type\_pool\_base< \_Tp, \_PoolTp, \_Thread >.

### 4.9.1 Detailed Description

```
template<typename _Tp, template< bool > class _PoolTp, bool _Thread>struct __gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp,
_Thread >
```

Policy for individual \_\_pool objects.

Definition at line 555 of file mt\_allocator.h.

The documentation for this struct was generated from the following file:

- [mt\\_allocator.h](#)

## 4.10 \_\_gnu\_cxx::\_\_pool< \_Thread > Class Template Reference

### 4.10.1 Detailed Description

```
template<bool _Thread>class __gnu_cxx::__pool< _Thread >
```

Data describing the underlying memory pool, parameterized on threading support.

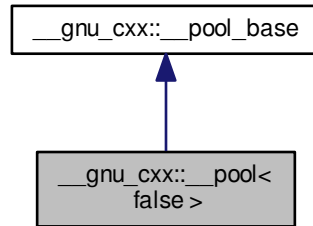
Definition at line 192 of file mt\_allocator.h.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

4.11 `__gnu_cxx::__pool< false >` Class Template Reference

Inheritance diagram for `__gnu_cxx::__pool< false >`:



## Public Types

- typedef unsigned short int **`_Binmap_type`**

## Public Member Functions

- **`__pool`** (const `__pool_base::_Tune &__tune`)
- void **`_M_adjust_freelist`** (const `_Bin_record &`, `_Block_record *`, `size_t`)
- bool **`_M_check_threshold`** (`size_t __bytes`)
- void **`_M_destroy`** () throw ()
- `size_t` **`_M_get_align`** ()
- const `_Bin_record &` **`_M_get_bin`** (`size_t __which`)
- `size_t` **`_M_get_binmap`** (`size_t __bytes`)
- const `_Tune &` **`_M_get_options`** () const
- `size_t` **`_M_get_thread_id`** ()
- void **`_M_initialize_once`** ()
- void **`_M_reclaim_block`** (`char *__p`, `size_t __bytes`) throw ()
- `char *` **`_M_reserve_block`** (`size_t __bytes`, const `size_t __thread_id`)
- void **`_M_set_options`** (`_Tune __t`)

## Protected Attributes

- `_Binmap_type *` **`_M_binmap`**
- bool **`_M_init`**
- `_Tune` **`_M_options`**

## 4.11.1 Detailed Description

```
template<>class __gnu_cxx::__pool< false >
```

Specialization for single thread.



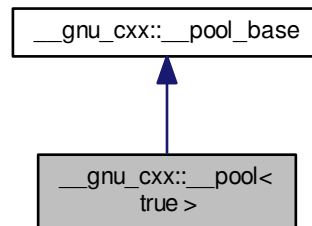
Definition at line 196 of file mt\_allocator.h.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

#### 4.12 \_\_gnu\_cxx::\_\_pool< true > Class Template Reference

Inheritance diagram for \_\_gnu\_cxx::\_\_pool< true >:



##### Public Types

- typedef unsigned short int **\_Binmap\_type**

##### Public Member Functions

- **\_\_pool** (const \_\_pool\_base::Tune &\_\_tune)
- void **\_M\_adjust\_freelist** (const \_Bin\_record &\_\_bin, \_Block\_record \* \_\_block, size\_t \_\_thread\_id)
- bool **\_M\_check\_threshold** (size\_t \_\_bytes)
- void **\_M\_destroy** () throw ()
- void **\_M\_destroy\_thread\_key** (void \*) throw ()
- size\_t **\_M\_get\_align** ()
- const \_Bin\_record & **\_M\_get\_bin** (size\_t \_\_which)
- size\_t **\_M\_get\_binmap** (size\_t \_\_bytes)
- const \_Tune & **\_M\_get\_options** () const
- size\_t **\_M\_get\_thread\_id** ()
- void **\_M\_initialize** (\_\_destroy\_handler)
- void **\_M\_initialize\_once** ()
- void **\_M\_reclaim\_block** (char \* \_\_p, size\_t \_\_bytes) throw ()
- char \* **\_M\_reserve\_block** (size\_t \_\_bytes, const size\_t \_\_thread\_id)
- void **\_M\_set\_options** (\_Tune \_\_t)

##### Protected Attributes

- \_Binmap\_type \* **\_M\_binmap**
- bool **\_M\_init**
- \_Tune **\_M\_options**

## 4.12.1 Detailed Description

```
template<>class __gnu_cxx::__pool< true >
```

Specialization for thread enabled, via gthreads.h.

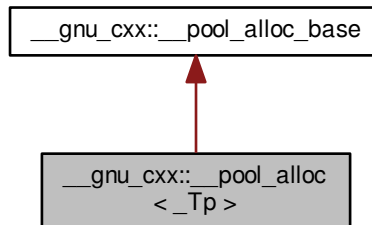
Definition at line 263 of file mt\_allocator.h.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

## 4.13 \_\_gnu\_cxx::\_\_pool\_alloc&lt;\_Tp&gt; Class Template Reference

Inheritance diagram for \_\_gnu\_cxx::\_\_pool\_alloc<\_Tp>:



## Public Types

- typedef const \_Tp \* **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef \_Tp \* **pointer**
- typedef std::true\_type **propagate\_on\_container\_move\_assignment**
- typedef \_Tp & **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- **\_\_pool\_alloc** (const [\\_\\_pool\\_alloc](#) &) noexcept
- template<typename \_Tp1 > **\_\_pool\_alloc** (const [\\_\\_pool\\_alloc](#)<\_Tp1> &) noexcept
- pointer **address** (reference \_\_x) const noexcept
- const\_pointer **address** (const\_reference \_\_x) const noexcept
- pointer **allocate** (size\_type \_\_n, const void \*==0)
- template<typename \_Up, typename... \_Args> void **construct** (\_Up \*\_\_p, \_Args &&...\_\_args)
- void **deallocate** (pointer \_\_p, size\_type \_\_n)
- template<typename \_Up > void **destroy** (\_Up \*\_\_p)
- size\_type **max\_size** () const noexcept

### Private Types

- enum { **\_S\_align** }
- enum { **\_S\_max\_bytes** }
- enum { **\_S\_free\_list\_size** }

### Private Member Functions

- char \* **\_M\_allocate\_chunk** (size\_t \_\_n, int &\_\_nobjs)
- \_Obj \*volatile \* **\_M\_get\_free\_list** (size\_t \_\_bytes) throw ()
- \_\_mutex & **\_M\_get\_mutex** () throw ()
- void \* **\_M\_refill** (size\_t \_\_n)
- size\_t **\_M\_round\_up** (size\_t \_\_bytes)

### Static Private Attributes

- static char \* **\_S\_end\_free**
- static \_Obj \*volatile **\_S\_free\_list** [\_S\_free\_list\_size]
- static size\_t **\_S\_heap\_size**
- static char \* **\_S\_start\_free**

#### 4.13.1 Detailed Description

template<typename \_Tp>class \_\_gnu\_cxx::\_\_pool\_alloc< \_Tp >

Allocator using a memory pool with a single lock.

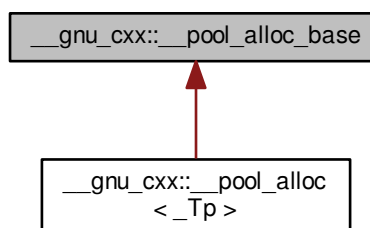
Definition at line 126 of file pool\_allocator.h.

The documentation for this class was generated from the following file:

- [pool\\_allocator.h](#)

#### 4.14 \_\_gnu\_cxx::\_\_pool\_alloc\_base Class Reference

Inheritance diagram for \_\_gnu\_cxx::\_\_pool\_alloc\_base:



### Protected Types

- enum { **\_S\_align** }
- enum { **\_S\_max\_bytes** }
- enum { **\_S\_free\_list\_size** }

### Protected Member Functions

- char \* **\_M\_allocate\_chunk** (size\_t \_\_n, int &\_\_nobjs)
- \_Obj \*volatile \* **\_M\_get\_free\_list** (size\_t \_\_bytes) throw ()
- \_\_mutex & **\_M\_get\_mutex** () throw ()
- void \* **\_M\_refill** (size\_t \_\_n)
- size\_t **\_M\_round\_up** (size\_t \_\_bytes)

### Static Protected Attributes

- static char \* **\_S\_end\_free**
- static \_Obj \*volatile **\_S\_free\_list** [\_S\_free\_list\_size]
- static size\_t **\_S\_heap\_size**
- static char \* **\_S\_start\_free**

#### 4.14.1 Detailed Description

Base class for `__pool_alloc`.

Uses various allocators to fulfill underlying requests (and makes as few requests as possible when in default high-speed pool mode).

Important implementation properties: 0. If globally mandated, then allocate objects from new 1. If the clients request an object of size  $> \_S\_max\_bytes$ , the resulting object will be obtained directly from new 2. In all other cases, we allocate an object of size exactly `\_S_round_up(requested_size)`. Thus the client has enough size information that we can return the object to the proper free list without permanently losing part of the object.

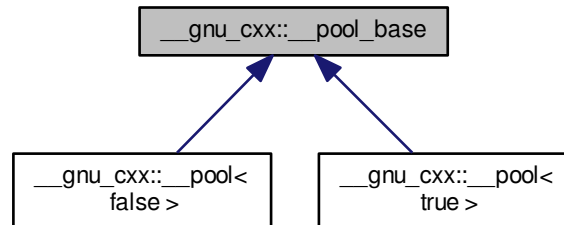
Definition at line 78 of file `pool_allocator.h`.

The documentation for this class was generated from the following file:

- [pool\\_allocator.h](#)

#### 4.15 `__gnu_cxx::__pool_base` Struct Reference

Inheritance diagram for `__gnu_cxx::__pool_base`:



##### Public Types

- typedef unsigned short int **`_Binmap_type`**

##### Public Member Functions

- **`__pool_base`** (const `_Tune` & `__options`)
- bool **`_M_check_threshold`** (size\_t `__bytes`)
- size\_t **`_M_get_align`** ()
- size\_t **`_M_get_binmap`** (size\_t `__bytes`)
- const `_Tune` & **`_M_get_options`** () const
- void **`_M_set_options`** (`_Tune` `__t`)

##### Protected Attributes

- `_Binmap_type` \* **`_M_binmap`**
- bool **`_M_init`**
- `_Tune` **`_M_options`**

##### 4.15.1 Detailed Description

Base class for pool object.

Definition at line 51 of file `mt_allocator.h`.

The documentation for this struct was generated from the following file:

- [mt\\_allocator.h](#)

#### 4.16 `__gnu_cxx::__rc_string_base<_CharT, _Traits, _Alloc>` Class Template Reference

Inherits `__gnu_cxx::__vstring_utility<_CharT, _Traits, _Alloc>`.

## Public Types

- `typedef _Util_Base::_CharT_alloc_type _CharT_alloc_type`
- `typedef __vstring_utility< _CharT, _Traits, _Alloc > _Util_Base`
- `typedef _Alloc allocator_type`
- `typedef _CharT_alloc_type::size_type size_type`
- `typedef _Traits traits_type`
- `typedef _Traits::char_type value_type`

## Public Member Functions

- `_rc_string_base (const _Alloc &__a)`
- `_rc_string_base (const \_\_rc\_string\_base &__rcs)`
- `_rc_string_base (\_\_rc\_string\_base &&__rcs)`
- `_rc_string_base (size_type __n, _CharT __c, const _Alloc &__a)`
- `template<typename _InputIterator > _rc_string_base (_InputIterator __beg, _InputIterator __end, const _Alloc &__a)`
- `void _M_assign (const \_\_rc\_string\_base &__rcs)`
- `size_type _M_capacity () const`
- `void _M_clear ()`
- `bool _M_compare (const \_\_rc\_string\_base &) const`
- `template<> bool _M_compare (const \_\_rc\_string\_base &__rcs) const`
- `template<> bool _M_compare (const \_\_rc\_string\_base &__rcs) const`
- `_CharT * _M_data () const`
- `void _M_erase (size_type __pos, size_type __n)`
- `allocator_type & _M_get_allocator ()`
- `const allocator_type & _M_get_allocator () const`
- `bool _M_is_shared () const`
- `void _M_leak ()`
- `size_type _M_length () const`
- `size_type _M_max_size () const`
- `void _M_mutate (size_type __pos, size_type __len1, const _CharT * __s, size_type __len2)`
- `void _M_reserve (size_type __res)`
- `void _M_set_leaked ()`
- `void _M_set_length (size_type __n)`
- `void _M_swap (\_\_rc\_string\_base &__rcs)`
- `template<typename _InIterator > _CharT * _S_construct (_InIterator __beg, _InIterator __end, const _Alloc &__a, std::forward\_iterator\_tag)`

## Protected Types

- `typedef __gnu_cxx::__normal_iterator< const_pointer, \_\_gnu\_cxx::\_\_versa\_string< _CharT, _Traits, _Alloc, \_\_rc\_string\_base > > __const_rc_iterator`
- `typedef __gnu_cxx::__normal_iterator< const_pointer, \_\_gnu\_cxx::\_\_versa\_string< _CharT, _Traits, _Alloc, \_\_sso\_string\_base > > __const_sso_iterator`
- `typedef __gnu_cxx::__normal_iterator< pointer, \_\_gnu\_cxx::\_\_versa\_string< _CharT, _Traits, _Alloc, \_\_rc\_string\_base > > __rc_iterator`
- `typedef __gnu_cxx::__normal_iterator< pointer, \_\_gnu\_cxx::\_\_versa\_string< _CharT, _Traits, _Alloc, \_\_sso\_string\_base > > __sso_iterator`
- `typedef _CharT_alloc_type::const_pointer const_pointer`
- `typedef _CharT_alloc_type::difference_type difference_type`
- `typedef _CharT_alloc_type::pointer pointer`

## Static Protected Member Functions

- static void **\_S\_assign** (\_CharT \*\_\_d, size\_type \_\_n, \_CharT \_\_c)
- static int **\_S\_compare** (size\_type \_\_n1, size\_type \_\_n2)
- static void **\_S\_copy** (\_CharT \*\_\_d, const \_CharT \*\_\_s, size\_type \_\_n)
- template<typename \_Iterator > static void **\_S\_copy\_chars** (\_CharT \*\_\_p, \_Iterator \_\_k1, \_Iterator \_\_k2)
- static void **\_S\_copy\_chars** (\_CharT \*\_\_p, \_\_sso\_iterator \_\_k1, \_\_sso\_iterator \_\_k2)
- static void **\_S\_copy\_chars** (\_CharT \*\_\_p, \_\_const\_sso\_iterator \_\_k1, \_\_const\_sso\_iterator \_\_k2)
- static void **\_S\_copy\_chars** (\_CharT \*\_\_p, \_\_rc\_iterator \_\_k1, \_\_rc\_iterator \_\_k2)
- static void **\_S\_copy\_chars** (\_CharT \*\_\_p, \_\_const\_rc\_iterator \_\_k1, \_\_const\_rc\_iterator \_\_k2)
- static void **\_S\_copy\_chars** (\_CharT \*\_\_p, \_CharT \*\_\_k1, \_CharT \*\_\_k2)
- static void **\_S\_copy\_chars** (\_CharT \*\_\_p, const \_CharT \*\_\_k1, const \_CharT \*\_\_k2)
- static void **\_S\_move** (\_CharT \*\_\_d, const \_CharT \*\_\_s, size\_type \_\_n)

### 4.16.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc> class __gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >
```

Documentation? What's that? Nathan Myers [ncm@cantrip.org](mailto:ncm@cantrip.org).

A string looks like this:

```

                                     [_Rep]
                                     _M_length
[ __rc_string_base<char_type>]      _M_capacity
_M_dataplus                        _M_refcount
_M_p ----->                      unnamed array of char_type
```

Where the `_M_p` points to the first character in the string, and you cast it to a pointer-to-`_Rep` and subtract 1 to get a pointer to the header.

This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single *add* instruction: `_Rep::_M_refdata()`, and `__rc_string_base::_M_rep()`; and the allocation function which gets a block of raw bytes and with room enough and constructs a `_Rep` object at the front.

The reason you want `_M_data` pointing to the character array and not the `_Rep` is so that the debugger can see the string contents. (Probably we should add a non-inline member to get the `_Rep` for the debugger to use, so users can check the actual string length.)

Note that the `_Rep` object is a POD so that you can have a static *empty string* `_Rep` object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string `_Rep` object.

All but the last paragraph is considered pretty conventional for a C++ string implementation.

Definition at line 82 of file `rc_string_base.h`.

The documentation for this class was generated from the following file:

- [rc\\_string\\_base.h](#)

## 4.17 \_\_gnu\_cxx::\_\_scoped\_lock Class Reference

### Public Types

- typedef \_\_mutex \_\_mutex\_type

## Public Member Functions

- `__scoped_lock` (`__mutex_type` & `__name`)

## 4.17.1 Detailed Description

Scoped lock idiom.

Definition at line 231 of file `concurrency.h`.

The documentation for this class was generated from the following file:

- [concurrency.h](#)

4.18 `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>` Class Template Reference

Inherits `_Base<_CharT, _Traits, _Alloc>`.

## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `__gnu_cxx::__normal_iterator< const_pointer, __versa_string >` **const\_iterator**
- typedef `_CharT_alloc_type::const_pointer` **const\_pointer**
- typedef `const value_type &` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_CharT_alloc_type::difference_type` **difference\_type**
- typedef `__gnu_cxx::__normal_iterator< pointer, __versa_string >` **iterator**
- typedef `_CharT_alloc_type::pointer` **pointer**
- typedef `value_type &` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_CharT_alloc_type::size_type` **size\_type**
- typedef `_Traits` **traits\_type**
- typedef `_Traits::char_type` **value\_type**

## Public Member Functions

- `__versa_string` ()
- `__versa_string` (const `_Alloc` & `__a`)
- `__versa_string` (const `__versa_string` & `__str`)
- `__versa_string` (`__versa_string` && `__str`) noexcept
- `__versa_string` (std::initializer\_list< `_CharT` > `__l`, const `_Alloc` & `__a`=`_Alloc`())
- `__versa_string` (const `__versa_string` & `__str`, size\_type `__pos`, size\_type `__n`=npos)
- `__versa_string` (const `__versa_string` & `__str`, size\_type `__pos`, size\_type `__n`, const `_Alloc` & `__a`)
- `__versa_string` (const `_CharT` \* `__s`, size\_type `__n`, const `_Alloc` & `__a`=`_Alloc`())
- `__versa_string` (const `_CharT` \* `__s`, const `_Alloc` & `__a`=`_Alloc`())
- `__versa_string` (size\_type `__n`, `_CharT` `__c`, const `_Alloc` & `__a`=`_Alloc`())
- template<class `_InputIterator` , typename = std::RequireInputIter<\_InputIterator>> `__versa_string` (`_InputIterator` `__beg`, `_InputIterator` `__end`, const `_Alloc` & `__a`=`_Alloc`())
- `~__versa_string` () noexcept
- `__versa_string` & `append` (const `__versa_string` & `__str`)



- [\\_\\_versa\\_string & append](#) (const [\\_\\_versa\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n)
- [\\_\\_versa\\_string & append](#) (const \_CharT \*\_\_s, size\_type \_\_n)
- [\\_\\_versa\\_string & append](#) (const \_CharT \*\_\_s)
- [\\_\\_versa\\_string & append](#) (size\_type \_\_n, \_CharT \_\_c)
- [\\_\\_versa\\_string & append](#) (std::initializer\_list< \_CharT > \_\_l)
- template<class \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>> [\\_\\_versa\\_string & append](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [\\_\\_versa\\_string & assign](#) (const [\\_\\_versa\\_string](#) &\_\_str)
- [\\_\\_versa\\_string & assign](#) ([\\_\\_versa\\_string](#) &&\_\_str)
- [\\_\\_versa\\_string & assign](#) (const [\\_\\_versa\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n)
- [\\_\\_versa\\_string & assign](#) (const \_CharT \*\_\_s, size\_type \_\_n)
- [\\_\\_versa\\_string & assign](#) (const \_CharT \*\_\_s)
- [\\_\\_versa\\_string & assign](#) (size\_type \_\_n, \_CharT \_\_c)
- template<class \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>> [\\_\\_versa\\_string & assign](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [\\_\\_versa\\_string & assign](#) (std::initializer\_list< \_CharT > \_\_l)
- const\_reference [at](#) (size\_type \_\_n) const
- reference [at](#) (size\_type \_\_n)
- reference [back](#) ()
- const\_reference [back](#) () const
- iterator [begin](#) () noexcept
- const\_iterator [begin](#) () const noexcept
- const \_CharT \* [c\\_str](#) () const noexcept
- size\_type [capacity](#) () const noexcept
- const\_iterator [cbegin](#) () const noexcept
- const\_iterator [cend](#) () const noexcept
- void [clear](#) () noexcept
- int [compare](#) (const [\\_\\_versa\\_string](#) &\_\_str) const
- int [compare](#) (size\_type \_\_pos, size\_type \_\_n, const [\\_\\_versa\\_string](#) &\_\_str) const
- int [compare](#) (size\_type \_\_pos1, size\_type \_\_n1, const [\\_\\_versa\\_string](#) &\_\_str, size\_type \_\_pos2, size\_type \_\_n2) const
- int [compare](#) (const \_CharT \*\_\_s) const
- int [compare](#) (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s) const
- int [compare](#) (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s, size\_type \_\_n2) const
- size\_type [copy](#) (\_CharT \*\_\_s, size\_type \_\_n, size\_type \_\_pos=0) const
- const\_reverse\_iterator [crbegin](#) () const noexcept
- const\_reverse\_iterator [crend](#) () const noexcept
- const \_CharT \* [data](#) () const noexcept
- bool [empty](#) () const noexcept
- iterator [end](#) () noexcept
- const\_iterator [end](#) () const noexcept
- [\\_\\_versa\\_string & erase](#) (size\_type \_\_pos=0, size\_type \_\_n=npos)
- iterator [erase](#) (iterator \_\_position)
- iterator [erase](#) (iterator \_\_first, iterator \_\_last)
- size\_type [find](#) (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type [find](#) (const [\\_\\_versa\\_string](#) &\_\_str, size\_type \_\_pos=0) const noexcept
- size\_type [find](#) (const \_CharT \*\_\_s, size\_type \_\_pos=0) const
- size\_type [find](#) (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- size\_type [find\\_first\\_not\\_of](#) (const [\\_\\_versa\\_string](#) &\_\_str, size\_type \_\_pos=0) const noexcept
- size\_type [find\\_first\\_not\\_of](#) (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type [find\\_first\\_not\\_of](#) (const \_CharT \*\_\_s, size\_type \_\_pos=0) const

- `size_type find_first_not_of` (`_CharT __c`, `size_type __pos=0`) `const noexcept`
- `size_type find_first_of` (`const __versa_string &__str`, `size_type __pos=0`) `const noexcept`
- `size_type find_first_of` (`const _CharT *__s`, `size_type __pos`, `size_type __n`) `const`
- `size_type find_first_of` (`const _CharT *__s`, `size_type __pos=0`) `const`
- `size_type find_first_of` (`_CharT __c`, `size_type __pos=0`) `const noexcept`
- `size_type find_last_not_of` (`const __versa_string &__str`, `size_type __pos=npos`) `const noexcept`
- `size_type find_last_not_of` (`const _CharT *__s`, `size_type __pos`, `size_type __n`) `const`
- `size_type find_last_not_of` (`const _CharT *__s`, `size_type __pos=npos`) `const`
- `size_type find_last_not_of` (`_CharT __c`, `size_type __pos=npos`) `const noexcept`
- `size_type find_last_of` (`const __versa_string &__str`, `size_type __pos=npos`) `const noexcept`
- `size_type find_last_of` (`const _CharT *__s`, `size_type __pos`, `size_type __n`) `const`
- `size_type find_last_of` (`const _CharT *__s`, `size_type __pos=npos`) `const`
- `size_type find_last_of` (`_CharT __c`, `size_type __pos=npos`) `const noexcept`
- reference `front` ()
- `const_reference front` () `const`
- `allocator_type get_allocator` () `const noexcept`
- `void insert` (`iterator __p`, `size_type __n`, `_CharT __c`)
- `template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>> void insert` (`iterator __p`, `_InputIterator __begin`, `_InputIterator __end`)
- `void insert` (`iterator __p`, `std::initializer_list<_CharT> __l`)
- `__versa_string &insert` (`size_type __pos1`, `const __versa_string &__str`)
- `__versa_string &insert` (`size_type __pos1`, `const __versa_string &__str`, `size_type __pos2`, `size_type __n`)
- `__versa_string &insert` (`size_type __pos`, `const _CharT *__s`, `size_type __n`)
- `__versa_string &insert` (`size_type __pos`, `const _CharT *__s`)
- `__versa_string &insert` (`size_type __pos`, `size_type __n`, `_CharT __c`)
- `iterator insert` (`iterator __p`, `_CharT __c`)
- `size_type length` () `const noexcept`
- `size_type max_size` () `const noexcept`
- `__versa_string &operator+=` (`const __versa_string &__str`)
- `__versa_string &operator+=` (`const _CharT *__s`)
- `__versa_string &operator+=` (`_CharT __c`)
- `__versa_string &operator+=` (`std::initializer_list<_CharT> __l`)
- `__versa_string &operator=` (`const __versa_string &__str`)
- `__versa_string &operator=` (`__versa_string &&__str`)
- `__versa_string &operator=` (`std::initializer_list<_CharT> __l`)
- `__versa_string &operator=` (`const _CharT *__s`)
- `__versa_string &operator=` (`_CharT __c`)
- `const_reference operator[]` (`size_type __pos`) `const`
- `reference operator[]` (`size_type __pos`)
- `void pop_back` ()
- `void push_back` (`_CharT __c`)
- `reverse_iterator rbegin` () `noexcept`
- `const_reverse_iterator rbegin` () `const noexcept`
- `reverse_iterator rend` () `noexcept`
- `const_reverse_iterator rend` () `const noexcept`
- `__versa_string &replace` (`size_type __pos`, `size_type __n`, `const __versa_string &__str`)
- `__versa_string &replace` (`size_type __pos1`, `size_type __n1`, `const __versa_string &__str`, `size_type __pos2`, `size_type __n2`)
- `__versa_string &replace` (`size_type __pos`, `size_type __n1`, `const _CharT *__s`, `size_type __n2`)
- `__versa_string &replace` (`size_type __pos`, `size_type __n1`, `const _CharT *__s`)
- `__versa_string &replace` (`size_type __pos`, `size_type __n1`, `size_type __n2`, `_CharT __c`)

- [\\_\\_versa\\_string & replace](#) (iterator \_\_i1, iterator \_\_i2, const [\\_\\_versa\\_string](#) &\_\_str)
- [\\_\\_versa\\_string & replace](#) (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_s, size\_type \_\_n)
- [\\_\\_versa\\_string & replace](#) (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_s)
- [\\_\\_versa\\_string & replace](#) (iterator \_\_i1, iterator \_\_i2, size\_type \_\_n, \_CharT \_\_c)
- template<class \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>> [\\_\\_versa\\_string & replace](#) (iterator \_\_i1, iterator \_\_i2, \_InputIterator \_\_k1, \_InputIterator \_\_k2)
- [\\_\\_versa\\_string & replace](#) (iterator \_\_i1, iterator \_\_i2, \_CharT \*\_\_k1, \_CharT \*\_\_k2)
- [\\_\\_versa\\_string & replace](#) (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_k1, const \_CharT \*\_\_k2)
- [\\_\\_versa\\_string & replace](#) (iterator \_\_i1, iterator \_\_i2, iterator \_\_k1, iterator \_\_k2)
- [\\_\\_versa\\_string & replace](#) (iterator \_\_i1, iterator \_\_i2, const\_iterator \_\_k1, const\_iterator \_\_k2)
- [\\_\\_versa\\_string & replace](#) (iterator \_\_i1, iterator \_\_i2, std::initializer\_list<\_CharT> \_\_l)
- void [reserve](#) (size\_type \_\_res\_arg=0)
- void [resize](#) (size\_type \_\_n, \_CharT \_\_c)
- void [resize](#) (size\_type \_\_n)
- size\_type [rfind](#) (const [\\_\\_versa\\_string](#) &\_\_str, size\_type \_\_pos=[npos](#)) const noexcept
- size\_type [rfind](#) (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type [rfind](#) (const \_CharT \*\_\_s, size\_type \_\_pos=[npos](#)) const
- size\_type [rfind](#) (\_CharT \_\_c, size\_type \_\_pos=[npos](#)) const noexcept
- void [shrink\\_to\\_fit](#) ()
- size\_type [size](#) () const noexcept
- [\\_\\_versa\\_string substr](#) (size\_type \_\_pos=0, size\_type \_\_n=[npos](#)) const
- void [swap](#) ([\\_\\_versa\\_string](#) &\_\_s)

#### Static Public Attributes

- static const size\_type [npos](#)

#### 4.18.1 Detailed Description

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

Template class [\\_\\_versa\\_string](#).

Data structure managing sequences of characters and character-like objects.

Definition at line 56 of file vstring.h.

#### 4.18.2 Constructor & Destructor Documentation

4.18.2.1 template<typename \_CharT, typename \_Traits, typename \_Alloc, template< typename, typename, typename > class  
[\\_Base](#)> [\\_\\_gnu\\_cxx::\\_\\_versa\\_string](#)< \_CharT, \_Traits, \_Alloc, \_Base >::[\\_\\_versa\\_string](#) ( ) [inline]

Default constructor creates an empty string.

Definition at line 134 of file vstring.h.

Referenced by [\\_\\_gnu\\_cxx::\\_\\_versa\\_string](#)< \_CharT, \_Traits, \_Alloc, \_Base >::substr().

4.18.2.2 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string ( const _Alloc & __a ) [inline], [explicit]`

Construct an empty string using allocator *a*.

Definition at line 141 of file `vstring.h`.

4.18.2.3 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string ( const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str ) [inline]`

Construct string with copy of value of `__str`.

Parameters

<code>__str</code>	Source string.
--------------------	----------------

Definition at line 149 of file `vstring.h`.

4.18.2.4 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string ( __versa_string<_CharT, _Traits, _Alloc, _Base> && __str ) [inline], [noexcept]`

String move constructor.

Parameters

<code>__str</code>	Source string.
--------------------	----------------

The newly-constructed string contains the exact contents of `__str`. The contents of `__str` are a valid, but unspecified string.

Definition at line 161 of file `vstring.h`.

4.18.2.5 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string ( std::initializer_list<_CharT> __l, const _Alloc & __a = _Alloc() ) [inline]`

Construct string from an initializer list.

Parameters

<code>__l</code>	<code>std::initializer_list</code> of characters.
<code>__a</code>	Allocator to use (default is default allocator).

Definition at line 169 of file `vstring.h`.

4.18.2.6 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string ( const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str, size_type __pos, size_type __n = npos ) [inline]`

Construct string as copy of a substring.

Parameters

<code>__str</code>	Source string.
--------------------	----------------

<code>__pos</code>	Index of first character to copy from.
<code>__n</code>	Number of characters to copy (default remainder).

Definition at line 180 of file `vstring.h`.

```
4.18.2.7 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename >
class _Base> __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string ( const
__versa_string< _CharT, _Traits, _Alloc, _Base > & __str, size_type __pos, size_type __n, const _Alloc & __a )
[inline]
```

Construct string as copy of a substring.

Parameters

<code>__str</code>	Source string.
<code>__pos</code>	Index of first character to copy from.
<code>__n</code>	Number of characters to copy.
<code>__a</code>	Allocator to use.

Definition at line 195 of file `vstring.h`.

```
4.18.2.8 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string ( const _CharT * __s,
size_type __n, const _Alloc & __a = _Alloc() ) [inline]
```

Construct string initialized by a character array.

Parameters

<code>__s</code>	Source character array.
<code>__n</code>	Number of characters to copy.
<code>__a</code>	Allocator to use (default is default allocator).

NB: `__s` must have at least `__n` characters, `'\0'` has no special meaning.

Definition at line 212 of file `vstring.h`.

```
4.18.2.9 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string ( const _CharT * __s,
const _Alloc & __a = _Alloc() ) [inline]
```

Construct string as copy of a C string.

Parameters

<code>__s</code>	Source C string.
<code>__a</code>	Allocator to use (default is default allocator).

Definition at line 221 of file `vstring.h`.

```
4.18.2.10 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string ( size_type __n, _CharT
__c, const _Alloc & __a = _Alloc() ) [inline]
```

Construct string as multiple characters.

## Parameters

<code>__n</code>	Number of characters.
<code>__c</code>	Character to use.
<code>__a</code>	Allocator to use (default is default allocator).

Definition at line 231 of file `vstring.h`.

```
4.18.2.11 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base> template<class _InputIterator, typename = std:: _RequireInputIter<_InputIterator>>
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string ( _InputIterator __beg,
_InputIterator __end, const _Alloc & __a = _Alloc() ) [inline]
```

Construct string as copy of a range.

## Parameters

<code>__beg</code>	Start of range.
<code>__end</code>	End of range.
<code>__a</code>	Allocator to use (default is default allocator).

Definition at line 246 of file `vstring.h`.

```
4.18.2.12 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::~~__versa_string ( ) [inline],
[noexcept]
```

Destroy the string instance.

Definition at line 253 of file `vstring.h`.

## 4.18.3 Member Function Documentation

```
4.18.3.1 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append ( const
__versa_string<_CharT, _Traits, _Alloc, _Base> & __str ) [inline]
```

Append a string to this string.

## Parameters

<code>__str</code>	The string to append.
--------------------	-----------------------

## Returns

Reference to this string.

Definition at line 688 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator+=( )`.

```
4.18.3.2 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append ( const
__versa_string<_CharT, _Traits, _Alloc, _Base> & __str, size_type __pos, size_type __n ) [inline]
```

Append a substring.

## Parameters

<code>__str</code>	The string to append.
<code>__pos</code>	Index of the first character of <code>str</code> to append.
<code>__n</code>	The number of characters to append.

## Returns

Reference to this string.

## Exceptions

<code>std::out_of_range</code>	if <code>pos</code> is not a valid index.
--------------------------------	---

This function appends `__n` characters from `__str` starting at `__pos` to this string. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is appended.

Definition at line 705 of file `vstring.h`.

```
4.18.3.3 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append ( const _CharT *
__s, size_type __n ) [inline]
```

Append a C substring.

## Parameters

<code>__s</code>	The C string to append.
<code>__n</code>	The number of characters to append.

## Returns

Reference to this string.

Definition at line 717 of file `vstring.h`.

```
4.18.3.4 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append ( const _CharT *
__s ) [inline]
```

Append a C string.

## Parameters

<code>__s</code>	The C string to append.
------------------	-------------------------

## Returns

Reference to this string.

Definition at line 730 of file `vstring.h`.

```
4.18.3.5 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append ( size_type __n,
_CharT __c ) [inline]
```

Append multiple characters.

## Parameters

<code>__n</code>	The number of characters to append.
<code>__c</code>	The character to use.

## Returns

Reference to this string.

Appends `n` copies of `c` to this string.

Definition at line 747 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

**4.18.3.6** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append ( std::initializer_list<_CharT > __l ) [inline]`

Append an `initializer_list` of characters.

## Parameters

<code>__l</code>	The <code>initializer_list</code> of characters to append.
------------------	--

## Returns

Reference to this string.

Definition at line 757 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append()`.

**4.18.3.7** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> template<class _InputIterator, typename = std:: RequireInputIter<_InputIterator>> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append ( _InputIterator __first, _InputIterator __last ) [inline]`

Append a range of characters.

## Parameters

<code>__first</code>	Iterator referencing the first character to append.
<code>__last</code>	Iterator marking the end of the range.

## Returns

Reference to this string.

Appends characters in the range `[first,last)` to this string.

Definition at line 776 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

**4.18.3.8** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign ( const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str ) [inline]`

Set value to contents of another string.



## Parameters

<code>__str</code>	Source string to use.
--------------------	-----------------------

## Returns

Reference to this string.

Definition at line 799 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator=()`.

```
4.18.3.9  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename >
          class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign (
          __versa_string<_CharT, _Traits, _Alloc, _Base> && __str ) [inline]
```

Set value to contents of another string.

## Parameters

<code>__str</code>	Source string to use.
--------------------	-----------------------

## Returns

Reference to this string.

This function sets this string to the exact contents of `__str`. `__str` is a valid, but unspecified string.

Definition at line 815 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::swap()`.

```
4.18.3.10 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
         _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign ( const
         __versa_string<_CharT, _Traits, _Alloc, _Base> & __str, size_type __pos, size_type __n ) [inline]
```

Set value to a substring of a string.

## Parameters

<code>__str</code>	The string to use.
<code>__pos</code>	Index of the first character of str.
<code>__n</code>	Number of characters to use.

## Returns

Reference to this string.

## Exceptions

<code>std::out_of_range</code>	if <code>__pos</code> is not a valid index.
--------------------------------	---

This function sets this string to the substring of `__str` consisting of `__n` characters at `__pos`. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is used.

Definition at line 836 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

```
4.18.3.11  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
           _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign ( const _CharT
           *__s, size_type __n )    [inline]
```

Set value to a C substring.

## Parameters

<code>__s</code>	The C string to use.
<code>__n</code>	Number of characters to use.

## Returns

Reference to this string.

This function sets the value of this string to the first `__n` characters of `__s`. If `__n` is larger than the number of available characters in `__s`, the remainder of `__s` is used.

Definition at line 853 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

```
4.18.3.12 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign ( const _CharT
*_s ) [inline]
```

Set value to contents of a C string.

## Parameters

<code>__s</code>	The C string to use.
------------------	----------------------

## Returns

Reference to this string.

This function sets the value of this string to the value of `__s`. The data is copied, so there is no dependence on `__s` once the function returns.

Definition at line 869 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

```
4.18.3.13 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign ( size_type __n,
_CharT __c ) [inline]
```

Set value to multiple characters.

## Parameters

<code>__n</code>	Length of the resulting string.
<code>__c</code>	The character to use.

## Returns

Reference to this string.

This function sets the value of this string to `__n` copies of character `__c`.

Definition at line 886 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

```
4.18.3.14  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
           _Base> template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>> __versa_string&
           __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign ( _InputIterator __first, _InputIterator __last )
           [inline]
```

Set value to a range of characters.

**Parameters**

<code>__first</code>	Iterator referencing the first character to append.
<code>__last</code>	Iterator marking the end of the range.

**Returns**

Reference to this string.

Sets value of string to characters in the range [first,last).

Definition at line 905 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

```
4.18.3.15  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename >
            class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign (
            std::initializer_list<_CharT> __l ) [inline]
```

Set value to an `initializer_list` of characters.

**Parameters**

<code>__l</code>	The <code>initializer_list</code> of characters to assign.
------------------	--

**Returns**

Reference to this string.

Definition at line 915 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign()`.

```
4.18.3.16  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
            _Base> const_reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::at ( size_type __n ) const
            [inline]
```

Provides access to the data contained in the string.

**Parameters**

<code>__n</code>	The index of the character to access.
------------------	---------------------------------------

**Returns**

Read-only (const) reference to the character.

**Exceptions**

<code>std::out_of_range</code>	If <code>__n</code> is an invalid index.
--------------------------------	--

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

Definition at line 579 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

```
4.18.3.17  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename >
            class _Base> reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::at ( size_type __n )
            [inline]
```

Provides access to the data contained in the string.

## Parameters

<code>__n</code>	The index of the character to access.
------------------	---------------------------------------

## Returns

Read/write reference to the character.

## Exceptions

<code>std::out_of_range</code>	If <code>__n</code> is an invalid index.
--------------------------------	--

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails. Success results in unsharing the string.

Definition at line 598 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**4.18.3.18** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::back ( ) [inline]`

Returns a read/write reference to the data at the last element of the string.

Definition at line 628 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator[]()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**4.18.3.19** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::back ( ) const [inline]`

Returns a read-only (constant) reference to the data at the last element of the string.

Definition at line 636 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator[]()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**4.18.3.20** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::begin ( ) [inline], [noexcept]`

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Definition at line 319 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::crend()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rend()`.

**4.18.3.21** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::begin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 330 of file `vstring.h`.

4.18.3.22 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const _CharT* __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::c_str ( ) const`  
`[inline], [noexcept]`

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1538 of file `vstring.h`.

4.18.3.23 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::capacity ( ) const`  
`[inline], [noexcept]`

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 490 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::push_back()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::shrink_to_fit()`.

4.18.3.24 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::cbegin ( ) const`  
`[inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 394 of file `vstring.h`.

4.18.3.25 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::cend ( ) const`  
`[inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 402 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

4.18.3.26 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::clear ( )` `[inline], [noexcept]`

Erases the string, making it empty.

Definition at line 518 of file `vstring.h`.

4.18.3.27 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare ( const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str ) const` `[inline]`

Compare to a string.

Parameters

<code>__str</code>	String to compare against.
--------------------	----------------------------



## Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before `__str`, 0 if their values are equivalent, or > 0 if this string is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 1964 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, `std::min()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::operator<()`, `__gnu_cxx::operator<=()`, `__gnu_cxx::operator==()`, `__gnu_cxx::operator>()`, and `__gnu_cxx::operator>=()`.

**4.18.3.28** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare ( size_type __pos, size_type __n, const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str ) const`

Compare substring to a string.

## Parameters

<code>__pos</code>	Index of first character of substring.
<code>__n</code>	Number of characters in substring.
<code>__str</code>	String to compare against.

## Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__str`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**4.18.3.29** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare ( size_type __pos1, size_type __n1, const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str, size_type __pos2, size_type __n2 ) const`

Compare substring to a substring.

## Parameters

<code>__pos1</code>	Index of first character of substring.
<code>__n1</code>	Number of characters in substring.
<code>__str</code>	String to compare against.
<code>__pos2</code>	Index of first character of substring of <code>str</code> .
<code>__n2</code>	Number of characters in substring of <code>str</code> .

## Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer < 0 if this substring is ordered before the substring of `__str`,

0 if their values are equivalent, or  $> 0$  if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(), str.substr(pos2, n2).data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

4.18.3.30 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare ( const _CharT * __s ) const`

Compare to a C string.

Parameters

<code>__s</code>	C string to compare against.
------------------	------------------------------

Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Returns an integer  $< 0$  if this string is ordered before `__s`,  $0$  if their values are equivalent, or  $> 0$  if this string is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and the length of a string constructed from `__s`. The function then compares the two strings by calling `traits::compare(data(), s, rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

4.18.3.31 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare ( size_type __pos, size_type __n1, const _CharT * __s ) const`

Compare substring to a C string.

Parameters

<code>__pos</code>	Index of first character of substring.
<code>__n1</code>	Number of characters in substring.
<code>__s</code>	C string to compare against.

Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n1` characters starting at `__pos`. Returns an integer  $< 0$  if the substring is ordered before `__s`,  $0$  if their values are equivalent, or  $> 0$  if the substring is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and the length of a string constructed from `__s`. The function then compares the two string by calling `traits::compare(substring.data(), s, rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

4.18.3.32 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare ( size_type __pos, size_type __n1, const _CharT * __s, size_type __n2 ) const`

Compare substring against a character array.

Parameters

<code>__pos</code>	Index of first character of substring.
<code>__n1</code>	Number of characters in substring.
<code>__s</code>	character array to compare against.
<code>__n2</code>	Number of characters of s.

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Form a string from the first `__n2` characters of `__s`. Returns an integer < 0 if this substring is ordered before the string from `__s`, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__n2`. The function then compares the two strings by calling `traits::compare(substring.data(), __s, rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: `__s` must have at least `n2` characters, `\0` has no special meaning.

```
4.18.3.33 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::copy ( _CharT * __s, size_type
__n, size_type __pos = 0 ) const
```

Copy substring into C string.

**Parameters**

<code>__s</code>	C string to copy value into.
<code>__n</code>	Number of characters to copy.
<code>__pos</code>	Index of first character to copy.

**Returns**

Number of characters actually copied

**Exceptions**

<code>std::out_of_range</code>	If <code>pos &gt; size()</code> .
--------------------------------	-----------------------------------

Copies up to `__n` characters starting at `__pos` into the C string `s`. If `__pos` is greater than `size()`, `out_of_range` is thrown.

```
4.18.3.34 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::crbegin ( )
const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 411 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end()`.

```
4.18.3.35 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::crend ( ) const
[inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 420 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::begin()`.

**4.18.336** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const _CharT* __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data ( ) const [inline], [noexcept]`

Return const pointer to contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1548 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_not_of()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of()`.

**4.18.337** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::empty ( ) const [inline], [noexcept]`

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 526 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**4.18.338** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::end ( ) [inline], [noexcept]`

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Definition at line 338 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::cbegin()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rbegin()`.

**4.18.339** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::end ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 349 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**4.18.340** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::erase ( size_type __pos = 0, size_type __n = npos ) [inline]`

Remove characters.

Parameters

---

<code>__pos</code>	Index of first character to remove (default 0).
<code>__n</code>	Number of characters to remove (default remainder).

**Returns**

Reference to this string.

**Exceptions**

<code>std::out_of_range</code>	If <code>__pos</code> is beyond the end of this string.
--------------------------------	---

Removes `__n` characters from this string starting at `__pos`. The length of the string is reduced by `__n`. If there are `< __n` characters to remove, the remainder of the string is truncated. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1115 of file `vstring.h`.

```
4.18.3.41 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase ( iterator __position )
[inline]
```

Remove one character.

**Parameters**

<code>__position</code>	Iterator referencing the character to remove.
-------------------------	---

**Returns**

iterator referencing same location after removal.

Removes the character at `__position` from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1131 of file `vstring.h`.

```
4.18.3.42 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase ( iterator __first, iterator
__last ) [inline]
```

Remove a range of characters.

**Parameters**

<code>__first</code>	Iterator referencing the first character to remove.
<code>__last</code>	Iterator referencing the end of the range.

**Returns**

Iterator referencing location of first after removal.

Removes the characters in the range `[first,last)` from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1152 of file `vstring.h`.

```
4.18.3.43 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find ( const _CharT * __s,
size_type __pos, size_type __n ) const
```

Find position of a C substring.

## Parameters

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to search from.
<code>__n</code>	Number of characters from <code>__s</code> to search for.

## Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_of()`.

4.18.3.44 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find ( const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str, size_type __pos = 0 ) const [inline], [noexcept]`

Find position of a string.

## Parameters

<code>__str</code>	String to locate.
<code>__pos</code>	Index of character to search from (default 0).

## Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1584 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::data()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

4.18.3.45 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find ( const _CharT * __s, size_type __pos = 0 ) const [inline]`

Find position of a C string.

## Parameters

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to search from (default 0).

## Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1599 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find()`.

4.18.3.46 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class  
_Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find ( _CharT __c, size_type  
__pos = 0 ) const [noexcept]`

Find position of a character.

## Parameters

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search from (default 0).

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

```
4.18.3.47 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename >
class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of (
const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str, size_type __pos = 0 ) const [inline],
[noexcept]
```

Find position of a character not in string.

## Parameters

<code>__str</code>	String containing characters to avoid.
<code>__pos</code>	Index of character to search from (default 0).

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1816 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::data()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_not_of()`.

```
4.18.3.48 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of ( const _CharT *
__s, size_type __pos, size_type __n ) const
```

Find position of a character not in C substring.

## Parameters

<code>__s</code>	C string containing characters to avoid.
<code>__pos</code>	Index of character to search from.
<code>__n</code>	Number of characters from <code>s</code> to consider.

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.



4.18.3.49 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class  
_Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of( const _CharT *  
__s, size_type __pos = 0 ) const [inline]`

Find position of a character not in C string.

## Parameters

<code>__s</code>	C string containing characters to avoid.
<code>__pos</code>	Index of character to search from (default 0).

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1847 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_not_of()`.

4.18.3.50 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_not_of ( _CharT __c, size_type __pos = 0 ) const [noexcept]`

Find position of a different character.

## Parameters

<code>__c</code>	Character to avoid.
<code>__pos</code>	Index of character to search from (default 0).

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

4.18.3.51 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_of ( const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str, size_type __pos = 0 ) const [inline], [noexcept]`

Find position of a character of string.

## Parameters

<code>__str</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search from (default 0).

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1689 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_of()`.

4.18.3.52 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class  
_Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of ( const _CharT *  
__s, size_type __pos, size_type __n ) const`

Find position of a character of C substring.

## Parameters

<code>__s</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search from.
<code>__n</code>	Number of characters from <code>s</code> to search for.

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

```
4.18.3.53 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_of ( const _CharT *
__s, size_type __pos = 0 ) const [inline]
```

Find position of a character of C string.

## Parameters

<code>__s</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search from (default 0).

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1719 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_of()`.

```
4.18.3.54 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_of ( _CharT __c,
size_type __pos = 0 ) const [inline], [noexcept]
```

Find position of a character.

## Parameters

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search from (default 0).

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for the character `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `find(c, pos)`.

Definition at line 1738 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find()`.

4.18.3.55 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class  
_Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of ( const  
__versa_string< _CharT, _Traits, _Alloc, _Base > & __str, size_type __pos = npos ) const [inline],  
[noexcept]`

Find last position of a character not in string.

## Parameters

<code>__str</code>	String containing characters to avoid.
<code>__pos</code>	Index of character to search back from (default end).

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1879 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_last_not_of()`.

**4.18.3.56** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_last_not_of ( const _CharT * __s, size_type __pos, size_type __n ) const`

Find last position of a character not in C substring.

## Parameters

<code>__s</code>	C string containing characters to avoid.
<code>__pos</code>	Index of character to search back from.
<code>__n</code>	Number of characters from <code>s</code> to consider.

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**4.18.3.57** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_last_not_of ( const _CharT * __s, size_type __pos = npos ) const [inline]`

Find last position of a character not in C string.

## Parameters

<code>__s</code>	C string containing characters to avoid.
<code>__pos</code>	Index of character to search back from (default end).

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1910 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_last_not_of()`.

4.18.3.58 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class  
_Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of ( _CharT __c,  
size_type __pos = npos ) const [noexcept]`

Find last position of a different character.

## Parameters

<code>__c</code>	Character to avoid.
<code>__pos</code>	Index of character to search back from (default end).

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

```
4.18.3.59 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename >
class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of( const
__versa_string< _CharT, _Traits, _Alloc, _Base > & __str, size_type __pos = npos ) const [inline],
[noexcept]
```

Find last position of a character of string.

## Parameters

<code>__str</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search back from (default end).

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1753 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of()`.

```
4.18.3.60 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of( const _CharT * __s,
size_type __pos, size_type __n ) const
```

Find last position of a character of C substring.

## Parameters

<code>__s</code>	C string containing characters to locate.
<code>__pos</code>	Index of character to search back from.
<code>__n</code>	Number of characters from <code>s</code> to search for.

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.



4.18.3.61 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class  
_Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of ( const _CharT * __s,  
size_type __pos = npos ) const [inline]`

Find last position of a character of C string.

## Parameters

<code>__s</code>	C string containing characters to locate.
<code>__pos</code>	Index of character to search back from (default end).

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1783 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_last_of()`.

```
4.18.3.62  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
           _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_last_of ( _CharT __c,
           size_type __pos = npos ) const    [inline], [noexcept]
```

Find last position of a character.

## Parameters

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search back from (default end).

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `rfind(c, pos)`.

Definition at line 1802 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::rfind()`.

```
4.18.3.63  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
           _Base> reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::front ( ) [inline]
```

Returns a read/write reference to the data at the first element of the string.

Definition at line 612 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator[]()`.

```
4.18.3.64  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
           _Base> const_reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::front ( ) const
           [inline]
```

Returns a read-only (constant) reference to the data at the first element of the string.

Definition at line 620 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator[]()`.

4.18.3.65 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> allocator_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::get_allocator ( ) const [inline], [noexcept]`

Return copy of allocator used to construct this string.

Definition at line 1555 of file `vstring.h`.

4.18.3.66 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert ( iterator __p, size_type __n, _CharT __c ) [inline]`

Insert multiple characters.

#### Parameters

<code>__p</code>	Iterator referencing location in string to insert at.
<code>__n</code>	Number of characters to insert
<code>__c</code>	The character to insert.

#### Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts `__n` copies of character `__c` starting at the position referenced by iterator `__p`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 933 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace()`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert()`.

4.18.3.67 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> template<class _InputIterator, typename = std::RequireInputIter<_InputIterator>> void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert ( iterator __p, _InputIterator __beg, _InputIterator __end ) [inline]`

Insert a range of characters.

#### Parameters

<code>__p</code>	Iterator referencing location in string to insert at.
<code>__beg</code>	Start of range.
<code>__end</code>	End of range.

#### Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts characters in range `[beg,end)`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 955 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace()`.

4.18.3.68 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert ( iterator __p, std::initializer_list< _CharT > __l ) [inline]`

Insert an `initializer_list` of characters.

## Parameters

<code>__p</code>	Iterator referencing location in string to insert at.
<code>__l</code>	The initializer_list of characters to insert.

## Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Definition at line 966 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert()`.

4.18.3.69 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert ( size_type __pos1, const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str ) [inline]`

Insert value of a string.

## Parameters

<code>__pos1</code>	Iterator referencing location in string to insert at.
<code>__str</code>	The string to insert.

## Returns

Reference to this string.

## Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts value of `__str` starting at `__pos1`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 983 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

4.18.3.70 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert ( size_type __pos1, const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str, size_type __pos2, size_type __n ) [inline]`

Insert a substring.

## Parameters

<code>__pos1</code>	Iterator referencing location in string to insert at.
<code>__str</code>	The string to insert.
<code>__pos2</code>	Start of characters in <code>str</code> to insert.
<code>__n</code>	Number of characters to insert.

## Returns

Reference to this string.

**Exceptions**

<i>std::length_error</i>	If new length exceeds <code>max_size()</code> .
<i>std::out_of_range</i>	If <code>__pos1 &gt; size()</code> or <code>__pos2 &gt; __str.size()</code> .

Starting at `__pos1`, insert `__n` character of `__str` beginning with `__pos2`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos1` is beyond the end of this string or `__pos2` is beyond the end of `__str`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1006 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

4.18.3.71 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert ( size_type __pos, const _CharT* __s, size_type __n ) [inline]`

Insert a C substring.

**Parameters**

<code>__pos</code>	Iterator referencing location in string to insert at.
<code>__s</code>	The C string to insert.
<code>__n</code>	The number of characters to insert.

**Returns**

Reference to this string.

**Exceptions**

<i>std::length_error</i>	If new length exceeds <code>max_size()</code> .
<i>std::out_of_range</i>	If <code>__pos</code> is beyond the end of this string.

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1029 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

4.18.3.72 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert ( size_type __pos, const _CharT* __s ) [inline]`

Insert a C string.

**Parameters**

<code>__pos</code>	Iterator referencing location in string to insert at.
<code>__s</code>	The C string to insert.

**Returns**

Reference to this string.

## Exceptions

<i>std::length_error</i>	If new length exceeds <code>max_size()</code> .
<i>std::out_of_range</i>	If <code>__pos</code> is beyond the end of this string.

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1048 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

4.18.3.73 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert ( size_type __pos, size_type __n, _CharT __c ) [inline]`

Insert multiple characters.

## Parameters

<code>__pos</code>	Index in string to insert at.
<code>__n</code>	Number of characters to insert
<code>__c</code>	The character to insert.

## Returns

Reference to this string.

## Exceptions

<i>std::length_error</i>	If new length exceeds <code>max_size()</code> .
<i>std::out_of_range</i>	If <code>__pos</code> is beyond the end of this string.

Inserts `__n` copies of character `__c` starting at index `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos > length()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1072 of file `vstring.h`.

4.18.3.74 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert ( iterator __p, _CharT __c ) [inline]`

Insert one character.

## Parameters

<code>__p</code>	Iterator referencing position in string to insert at.
<code>__c</code>	The character to insert.

## Returns

Iterator referencing newly inserted char.

## Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts character `__c` at position referenced by `__p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1090 of file `vstring.h`.

4.18.3.75 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::length ( ) const [inline], [noexcept]`

Returns the number of characters in the string, not including any null-termination.

Definition at line 435 of file `vstring.h`.

4.18.3.76 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::max_size ( ) const [inline], [noexcept]`

Returns the `size()` of the largest possible string.

Definition at line 440 of file `vstring.h`.

4.18.3.77 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+=( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str ) [inline]`

Append a string to this string.

## Parameters

<code>__str</code>	The string to append.
--------------------	-----------------------

## Returns

Reference to this string.

Definition at line 647 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`.

4.18.3.78 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+=( const _CharT * __s ) [inline]`

Append a C string.

## Parameters

<code>__s</code>	The C string to append.
------------------	-------------------------

## Returns

Reference to this string.

Definition at line 656 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`.

4.18.3.79 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator+=( _CharT __c ) [inline]`

Append a character.



## Parameters

<code>__c</code>	The character to append.
------------------	--------------------------

## Returns

Reference to this string.

Definition at line 665 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::push_back()`.

4.18.3.80 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator+=( std::initializer_list<_CharT> __l ) [inline]`

Append an `initializer_list` of characters.

## Parameters

<code>__l</code>	The <code>initializer_list</code> of characters to be appended.
------------------	---

## Returns

Reference to this string.

Definition at line 678 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`.

4.18.3.81 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator= ( const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str ) [inline]`

Assign the value of `str` to this string.

## Parameters

<code>__str</code>	Source string.
--------------------	----------------

Definition at line 260 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign()`.

4.18.3.82 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator= ( __versa_string<_CharT, _Traits, _Alloc, _Base> && __str ) [inline]`

String move assignment operator.

## Parameters

<code>__str</code>	Source string.
--------------------	----------------

The contents of `__str` are moved into this string (without copying). `__str` is a valid, but unspecified string.

Definition at line 272 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::swap()`.

```
4.18.3.83  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename >
            class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator= (
            std::initializer_list<_CharT> __l ) [inline]
```

Set value to string constructed from initializer list.

## Parameters

<code>__l</code>	std::initializer_list.
------------------	------------------------

Definition at line 284 of file vstring.h.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign()`.

4.18.3.84 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator= ( const _CharT* __s ) [inline]`

Copy contents of `__s` into this string.

## Parameters

<code>__s</code>	Source null-terminated string.
------------------	--------------------------------

Definition at line 296 of file vstring.h.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign()`.

4.18.3.85 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator= ( _CharT __c ) [inline]`

Set value to string of length 1.

## Parameters

<code>__c</code>	Source character.
------------------	-------------------

Assigning to a character makes this string length 1 and `(*this)[0] == __c`.

Definition at line 307 of file vstring.h.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign()`.

4.18.3.86 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator[] ( size_type __pos ) const [inline]`

Subscript access to the data contained in the string.

## Parameters

<code>__pos</code>	The index of the character to access.
--------------------	---------------------------------------

## Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 541 of file vstring.h.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::back()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::front()`.

```
4.18.3.87  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
           _Base> reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator[]( size_type __pos )
           [inline]
```

Subscript access to the data contained in the string.

## Parameters

<code>__pos</code>	The index of the character to access.
--------------------	---------------------------------------

## Returns

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.) Unshares the string.

Definition at line 558 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**4.18.3.88** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::pop_back ( ) [inline]`

Remove the last character.

The string must be non-empty.

Definition at line 1169 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**4.18.3.89** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::push_back ( _CharT __c ) [inline]`

Append a single character.

## Parameters

<code>__c</code>	Character to append.
------------------	----------------------

Definition at line 784 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::capacity()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator+=( )`.

**4.18.3.90** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> reverse_iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rbegin ( ) [inline], [noexcept]`

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 358 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::end()`.

**4.18.3.91** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reverse_iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 367 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::end()`.

4.18.3.92 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> reverse_iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::rend ( )`  
`[inline], [noexcept]`

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 376 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::begin()`.

4.18.3.93 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reverse_iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::rend ( ) const`  
`[inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 385 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::begin()`.

4.18.3.94 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace ( size_type __pos, size_type __n, const __versa_string<_CharT, _Traits, _Alloc, _Base > &__str )` `[inline]`

Replace characters with value from another string.

#### Parameters

<code>__pos</code>	Index of first character to replace.
<code>__n</code>	Number of characters to be replaced.
<code>__str</code>	String to insert.

#### Returns

Reference to this string.

#### Exceptions

<code>std::out_of_range</code>	If <code>__pos</code> is beyond the end of this string.
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[pos, pos+n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1191 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::insert()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

4.18.3.95 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class  
_Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace ( size_type  
__pos1, size_type __n1, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str, size_type __pos2, size_type  
__n2 ) [inline]`

Replace characters with value from another string.

## Parameters

<code>__pos1</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__str</code>	String to insert.
<code>__pos2</code>	Index of first character of <code>str</code> to use.
<code>__n2</code>	Number of characters from <code>str</code> to use.

## Returns

Reference to this string.

## Exceptions

<code>std::out_of_range</code>	If <code>__pos1 &gt; size()</code> or <code>__pos2 &gt; str.size()</code> .
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[pos1, pos1 + n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1214 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

4.18.3.96 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace ( size_type __pos, size_type __n1, const _CharT* __s, size_type __n2 ) [inline]`

Replace characters with value of a C substring.

## Parameters

<code>__pos</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__s</code>	C string to insert.
<code>__n2</code>	Number of characters from <code>__s</code> to use.

## Returns

Reference to this string.

## Exceptions

<code>std::out_of_range</code>	If <code>__pos1 &gt; size()</code> .
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[pos, pos + n1)` from this string. In place, the first `__n2` characters of `__s` are inserted, or all of `__s` if `__n2` is too large. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1242 of file `vstring.h`.

4.18.3.97 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace ( size_type __pos, size_type __n1, const _CharT* __s ) [inline]`

Replace characters with value of a C string.



## Parameters

<code>__pos</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__s</code>	C string to insert.

## Returns

Reference to this string.

## Exceptions

<code>std::out_of_range</code>	If <code>__pos &gt; size()</code> .
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[pos, pos + n1)` from this string. In place, the characters of `__s` are inserted. If `pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1266 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

4.18.3.98 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace ( size_type __pos, size_type __n1, size_type __n2, _CharT __c ) [inline]`

Replace characters with multiple characters.

## Parameters

<code>__pos</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__n2</code>	Number of characters to insert.
<code>__c</code>	Character to insert.

## Returns

Reference to this string.

## Exceptions

<code>std::out_of_range</code>	If <code>__pos &gt; size()</code> .
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[pos, pos + n1)` from this string. In place, `__n2` copies of `__c` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1290 of file `vstring.h`.

4.18.3.99 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace ( iterator __i1, iterator __i2, const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str ) [inline]`

Replace range of characters with string.

## Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__str</code>	String value to insert.

## Returns

Reference to this string.

## Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[i1,i2)`. In place, the value of `__str` is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1308 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

4.18.3.100 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace ( iterator __i1, iterator __i2, const _CharT* __s, size_type __n ) [inline]`

Replace range of characters with C substring.

## Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__s</code>	C string value to insert.
<code>__n</code>	Number of characters from <code>s</code> to insert.

## Returns

Reference to this string.

## Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[i1,i2)`. In place, the first `n` characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1326 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

4.18.3.101 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace ( iterator __i1, iterator __i2, const _CharT* __s ) [inline]`

Replace range of characters with C string.

**Parameters**

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__s</code>	C string value to insert.

**Returns**

Reference to this string.

**Exceptions**

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[i1,i2)`. In place, the characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1347 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

4.18.3.102 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace ( iterator __i1, iterator __i2, size_type __n, _CharT __c ) [inline]`

Replace range of characters with multiple characters.

**Parameters**

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__n</code>	Number of characters to insert.
<code>__c</code>	Character to insert.

**Returns**

Reference to this string.

**Exceptions**

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[i1,i2)`. In place, `__n` copies of `__c` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1368 of file `vstring.h`.

4.18.3.103 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> template<class _InputIterator, typename = std:: RequireInputIter<_InputIterator>> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace ( iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2 ) [inline]`

Replace range of characters with range.

**Parameters**

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__k1</code>	Iterator referencing start of range to insert.
<code>__k2</code>	Iterator referencing end of range to insert.

**Returns**

Reference to this string.

**Exceptions**

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[i1,i2)`. In place, characters in the range `[k1,k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1393 of file `vstring.h`.

4.18.3.104 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace ( iterator __i1, iterator __i2, std::initializer_list<_CharT > __l ) [inline]`

Replace range of characters with `initializer_list`.

**Parameters**

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__l</code>	The <code>initializer_list</code> of characters to insert.

**Returns**

Reference to this string.

**Exceptions**

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[i1,i2)`. In place, characters in the range `[k1,k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1474 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

4.18.3.105 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::reserve ( size_type __res_arg = 0 ) [inline]`

Attempt to preallocate enough memory for specified number of characters.

**Parameters**

<code>__res_arg</code>	Number of characters required.
------------------------	--------------------------------

## Exceptions

<i>std::length_error</i>	If <code>__res_arg</code> exceeds <code>max_size()</code> .
--------------------------	---

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

Definition at line 511 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::shrink_to_fit()`.

4.18.3.106 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::resize ( size_type __n, _CharT __c )`

Resizes the string to the specified number of characters.

## Parameters

<code>__n</code>	Number of characters the string should contain.
<code>__c</code>	Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to `__c`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::resize()`.

4.18.3.107 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::resize ( size_type __n ) [inline]`

Resizes the string to the specified number of characters.

## Parameters

<code>__n</code>	Number of characters the string should contain.
------------------	---

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as `char`, this means setting them to 0.

Definition at line 467 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::resize()`.

4.18.3.108 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rfind ( const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str, size_type __pos = npos ) const [inline], [noexcept]`

Find last position of a string.

## Parameters

<code>__str</code>	String to locate.
<code>__pos</code>	Index of character to search back from (default end).

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1629 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind()`.

4.18.3.109 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind ( const _CharT * __s, size_type __pos, size_type __n ) const`

Find last position of a C substring.

**Parameters**

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to search back from.
<code>__n</code>	Number of characters from <code>s</code> to search for.

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

4.18.3.110 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind ( const _CharT * __s, size_type __pos = npos ) const [inline]`

Find last position of a C string.

**Parameters**

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to start search at (default end).

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1659 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind()`.

4.18.3.111 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind ( _CharT __c, size_type __pos = npos ) const [noexcept]`

Find last position of a character.

## Parameters

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search back from (default end).

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

4.18.3.112 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::shrink_to_fit ( ) [inline]`

A non-binding request to reduce capacity() to size().

Definition at line 473 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::capacity()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::reserve()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

4.18.3.113 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size ( ) const [inline], [noexcept]`

Returns the number of characters in the string, not including any null-termination.

Definition at line 429 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::at()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::back()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::cend()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::empty()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator[]()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::pop_back()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::push_back()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::shrink_to_fit()`.

4.18.3.114 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::substr ( size_type __pos = 0, size_type __n = npow ) const [inline]`

Get a substring.

## Parameters

<code>__pos</code>	Index of first character (default 0).
<code>__n</code>	Number of characters in substring (default remainder).

## Returns

The new string.

## Exceptions

<code>std::out_of_range</code>	If <code>pos &gt; size()</code> .
--------------------------------	-----------------------------------

Construct and return a new string using the `__n` characters starting at `__pos`. If the string is too short, use the remainder of the characters. If `__pos` is beyond the end of the string, `out_of_range` is thrown.

Definition at line 1943 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string()`.

4.18.3.115 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::swap ( __versa_string<_CharT, _Traits, _Alloc, _Base> & __s ) [inline]`

Swap contents with another string.

## Parameters

<code>__s</code>	String to swap with.
------------------	----------------------

Exchanges the contents of this string with that of `__s` in constant time.

Definition at line 1527 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator=()`, and `__gnu_cxx::swap()`.

## 4.18.4 Member Data Documentation

4.18.4.1 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos [static]`

Value returned by various member functions when they fail.

Definition at line 81 of file `vstring.h`.

The documentation for this class was generated from the following file:

- [vstring.h](#)

4.19 `__gnu_cxx::Caster<_ToType>` Struct Template Reference

## Public Types

- `typedef _ToType::element_type * type`

## 4.19.1 Detailed Description

`template<typename _ToType> struct __gnu_cxx::Caster<_ToType>`

These functions are here to allow containers to support non standard pointer types. For normal pointers, these resolve to the use of the standard cast operation. For other types the functions will perform the appropriate cast to/from the custom pointer class so long as that class meets the following conditions: 1) has a typedef `element_type` which names the type it points to. 2) has a `get()` const method which returns `element_type*`. 3) has a constructor which can take one `element_type*` argument. This type supports the semantics of the pointer cast operators (below.)

Definition at line 52 of file `cast.h`.



The documentation for this struct was generated from the following file:

- [cast.h](#)

## 4.20 `__gnu_cxx::Char_types<_CharT>` Struct Template Reference

### Public Types

- typedef unsigned long **int\_type**
- typedef [std::streamoff](#) **off\_type**
- typedef [std::streampos](#) **pos\_type**
- typedef [std::mbstate\\_t](#) **state\_type**

### 4.20.1 Detailed Description

```
template<typename _CharT>struct __gnu_cxx::Char_types<_CharT>
```

Mapping from character type to associated types.

### Note

This is an implementation class for the generic version of `char_traits`. It defines `int_type`, `off_type`, `pos_type`, and `state_type`. By default these are unsigned long, streamoff, streampos, and mbstate\_t. Users who need a different set of types, but who don't need to change the definitions of any function defined in `char_traits`, can specialize `__gnu_cxx::Char_types` while leaving `__gnu_cxx::char_traits` alone.

Definition at line 58 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 4.21 `__gnu_cxx::ExtPtr_allocator<_Tp>` Class Template Reference

### Public Types

- typedef [\\_Pointer\\_adapter](#)<[\\_Relative\\_pointer\\_impl](#)< const \_Tp > > **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef [std::ptrdiff\\_t](#) **difference\_type**
- typedef [\\_Pointer\\_adapter](#)<[\\_Relative\\_pointer\\_impl](#)< \_Tp > > **pointer**
- typedef \_Tp & **reference**
- typedef [std::size\\_t](#) **size\_type**
- typedef \_Tp **value\_type**

### Public Member Functions

- **\_ExtPtr\_allocator** (const [\\_ExtPtr\\_allocator](#) &\_\_rarg) noexcept
- template<typename \_Up > **\_ExtPtr\_allocator** (const [\\_ExtPtr\\_allocator](#)< \_Up > &\_\_rarg) noexcept
- const [std::allocator](#)< \_Tp > & **\_M\_getUnderlyingImp** () const
- **pointer address** (reference \_\_x) const noexcept
- **const\_pointer address** (const\_reference \_\_x) const noexcept

- [pointer allocate](#) (size\_type \_\_n, void \*\_\_hint=0)
- template<typename \_Up, typename... \_Args> void **construct** (\_Up \*\_\_p, \_Args &&... \_\_args)
- template<typename... \_Args> void **construct** ([pointer](#) \_\_p, \_Args &&... \_\_args)
- void **deallocate** ([pointer](#) \_\_p, size\_type \_\_n)
- template<typename \_Up> void **destroy** (\_Up \*\_\_p)
- void **destroy** ([pointer](#) \_\_p)
- size\_type **max\_size** () const noexcept
- template<typename \_Up> bool **operator!=** (const [\\_ExtPtr\\_allocator](#)<\_Up> &\_\_rarg)
- bool **operator!=** (const [\\_ExtPtr\\_allocator](#) &\_\_rarg)
- template<typename \_Up> bool **operator==** (const [\\_ExtPtr\\_allocator](#)<\_Up> &\_\_rarg)
- bool **operator==** (const [\\_ExtPtr\\_allocator](#) &\_\_rarg)

#### Friends

- template<typename \_Up> void **swap** ([\\_ExtPtr\\_allocator](#)<\_Up> &, [\\_ExtPtr\\_allocator](#)<\_Up> &)

#### 4.21.1 Detailed Description

template<typename \_Tp>class `__gnu_cxx::_ExtPtr_allocator`<\_Tp>

An example allocator which uses a non-standard pointer type.

This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See `ext/pointer.h`) Memory allocation in this example is still performed using `std::allocator`.

Definition at line 56 of file `extptr_allocator.h`.

The documentation for this class was generated from the following file:

- [extptr\\_allocator.h](#)

## 4.22 `__gnu_cxx::_Invalid_type` Struct Reference

#### 4.22.1 Detailed Description

The specialization on this type helps resolve the problem of reference to void, and eliminates the need to specialize `_Pointer_adapter` for cases of `void*`, `const void*`, and so on.

Definition at line 213 of file `pointer.h`.

The documentation for this struct was generated from the following file:

- [pointer.h](#)

## 4.23 `__gnu_cxx::_Pointer_adapter`<`_Storage_policy`> Class Template Reference

Inherits `_Storage_policy`.

## Public Types

- typedef std::ptrdiff\_t **difference\_type**
- typedef \_Storage\_policy::element\_type **element\_type**
- typedef [std::random\\_access\\_iterator\\_tag](#) **iterator\_category**
- typedef [\\_Pointer\\_adapter](#) **pointer**
- typedef \_Reference\_type< element\_type >::reference **reference**
- typedef [\\_Unqualified\\_type](#)< element\_type >::type **value\_type**

## Public Member Functions

- [\\_Pointer\\_adapter](#) (element\_type \*\_\_arg=0)
- [\\_Pointer\\_adapter](#) (const [\\_Pointer\\_adapter](#) &\_\_arg)
- template<typename \_Up > [\\_Pointer\\_adapter](#) (\_Up \*\_\_arg)
- template<typename \_Up > [\\_Pointer\\_adapter](#) (const [\\_Pointer\\_adapter](#)< \_Up > &\_\_arg)
- **operator \_\_unspecified\_bool\_type** () const
- bool **operator!** () const
- reference **operator\*** () const
- [\\_Pointer\\_adapter](#) & **operator++** ()
- [\\_Pointer\\_adapter](#) **operator++** (int)
- [\\_Pointer\\_adapter](#) & **operator+=** (short \_\_offset)
- [\\_Pointer\\_adapter](#) & **operator+=** (unsigned short \_\_offset)
- [\\_Pointer\\_adapter](#) & **operator+=** (int \_\_offset)
- [\\_Pointer\\_adapter](#) & **operator+=** (unsigned int \_\_offset)
- [\\_Pointer\\_adapter](#) & **operator+=** (long \_\_offset)
- [\\_Pointer\\_adapter](#) & **operator+=** (unsigned long \_\_offset)
- template<typename \_Up > std::ptrdiff\_t **operator-** (const [\\_Pointer\\_adapter](#)< \_Up > &\_\_rhs) const
- [\\_Pointer\\_adapter](#) & **operator--** ()
- [\\_Pointer\\_adapter](#) **operator--** (int)
- [\\_Pointer\\_adapter](#) & **operator-=** (short \_\_offset)
- [\\_Pointer\\_adapter](#) & **operator-=** (unsigned short \_\_offset)
- [\\_Pointer\\_adapter](#) & **operator-=** (int \_\_offset)
- [\\_Pointer\\_adapter](#) & **operator-=** (unsigned int \_\_offset)
- [\\_Pointer\\_adapter](#) & **operator-=** (long \_\_offset)
- [\\_Pointer\\_adapter](#) & **operator-=** (unsigned long \_\_offset)
- element\_type \* **operator->** () const
- [\\_Pointer\\_adapter](#) & **operator=** (const [\\_Pointer\\_adapter](#) &\_\_arg)
- template<typename \_Up > [\\_Pointer\\_adapter](#) & **operator=** (const [\\_Pointer\\_adapter](#)< \_Up > &\_\_arg)
- template<typename \_Up > [\\_Pointer\\_adapter](#) & **operator=** (\_Up \*\_\_arg)
- reference **operator[]** (std::ptrdiff\_t \_\_index) const

## Friends

- [\\_Pointer\\_adapter](#) **operator+** (const [\\_Pointer\\_adapter](#) &\_\_lhs, short \_\_offset)
- [\\_Pointer\\_adapter](#) **operator+** (short \_\_offset, const [\\_Pointer\\_adapter](#) &\_\_rhs)
- [\\_Pointer\\_adapter](#) **operator+** (const [\\_Pointer\\_adapter](#) &\_\_lhs, unsigned short \_\_offset)
- [\\_Pointer\\_adapter](#) **operator+** (unsigned short \_\_offset, const [\\_Pointer\\_adapter](#) &\_\_rhs)
- [\\_Pointer\\_adapter](#) **operator+** (const [\\_Pointer\\_adapter](#) &\_\_lhs, int \_\_offset)
- [\\_Pointer\\_adapter](#) **operator+** (int \_\_offset, const [\\_Pointer\\_adapter](#) &\_\_rhs)
- [\\_Pointer\\_adapter](#) **operator+** (const [\\_Pointer\\_adapter](#) &\_\_lhs, unsigned int \_\_offset)

- `_Pointer_adapter operator+` (unsigned int \_\_offset, const `_Pointer_adapter` &\_\_rhs)
- `_Pointer_adapter operator+` (const `_Pointer_adapter` &\_\_lhs, long \_\_offset)
- `_Pointer_adapter operator+` (long \_\_offset, const `_Pointer_adapter` &\_\_rhs)
- `_Pointer_adapter operator+` (unsigned long \_\_offset, const `_Pointer_adapter` &\_\_rhs)
- `_Pointer_adapter operator+` (const `_Pointer_adapter` &\_\_lhs, unsigned long \_\_offset)
- `std::ptrdiff_t operator-` (const `_Pointer_adapter` &\_\_lhs, element\_type \*\_\_rhs)
- `std::ptrdiff_t operator-` (element\_type \*\_\_lhs, const `_Pointer_adapter` &\_\_rhs)
- `template<typename _Up> std::ptrdiff_t operator-` (const `_Pointer_adapter` &\_\_lhs, \_Up \*\_\_rhs)
- `template<typename _Up> std::ptrdiff_t operator-` (\_Up \*\_\_lhs, const `_Pointer_adapter` &\_\_rhs)
- `_Pointer_adapter operator-` (const `_Pointer_adapter` &\_\_lhs, short \_\_offset)
- `_Pointer_adapter operator-` (const `_Pointer_adapter` &\_\_lhs, unsigned short \_\_offset)
- `_Pointer_adapter operator-` (const `_Pointer_adapter` &\_\_lhs, int \_\_offset)
- `_Pointer_adapter operator-` (const `_Pointer_adapter` &\_\_lhs, unsigned int \_\_offset)
- `_Pointer_adapter operator-` (const `_Pointer_adapter` &\_\_lhs, long \_\_offset)
- `_Pointer_adapter operator-` (const `_Pointer_adapter` &\_\_lhs, unsigned long \_\_offset)

#### 4.23.1 Detailed Description

```
template<typename _Storage_policy> class __gnu_cxx::Pointer_adapter<_Storage_policy>
```

The following provides an 'alternative pointer' that works with the containers when specified as the pointer typedef of the allocator.

The pointer type used with the containers doesn't have to be this class, but it must support the implicit conversions, pointer arithmetic, comparison operators, etc. that are supported by this class, and avoid raising compile-time ambiguities. Because creating a working pointer can be challenging, this pointer template was designed to wrapper an easier storage policy type, so that it becomes reusable for creating other pointer types.

A key point of this class is also that it allows container writers to 'assume' `Allocator::pointer` is a typedef for a normal pointer. This class supports most of the conventions of a true pointer, and can, for instance handle implicit conversion to const and base class pointer types. The only impositions on container writers to support extended pointers are: 1) use the `Allocator::pointer` typedef appropriately for pointer types. 2) if you need pointer casting, use the `__pointer_cast<>` functions from `ext/cast.h`. This allows pointer cast operations to be overloaded as necessary by custom pointers.

Note: The const qualifier works with this pointer adapter as follows:

```
_Tp* == _Pointer_adapter<_Std_pointer_impl<_Tp>>; const _Tp* == _Pointer_adapter<_Std_pointer_impl<const
_Tp>>; _Tp* const == const _Pointer_adapter<_Std_pointer_impl<_Tp>>; const _Tp* const == const _Pointer_
adapter<_Std_pointer_impl<const _Tp>>;
```

Definition at line 281 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

## 4.24 `__gnu_cxx::Relative_pointer_impl<_Tp>` Class Template Reference

### Public Types

- typedef `_Tp element_type`

### Public Member Functions

- `_Tp * get () const`
- `bool operator< (const \_Relative\_pointer\_impl &__rarg) const`
- `bool operator== (const \_Relative\_pointer\_impl &__rarg) const`
- `void set (_Tp *__arg)`

#### 4.24.1 Detailed Description

```
template<typename _Tp>class __gnu_cxx::_Relative_pointer_impl< _Tp >
```

A storage policy for use with `_Pointer_adapter<>` which stores the pointer's address as an offset value which is relative to its own address.

This is intended for pointers within shared memory regions which might be mapped at different addresses by different processes. For null pointers, a value of 1 is used. (0 is legitimate sometimes for nodes in circularly linked lists) This value was chosen as the least likely to generate an incorrect null, As there is no reason why any normal pointer would point 1 byte into its own pointer address.

Definition at line 109 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

## 4.25 `__gnu_cxx::_Relative_pointer_impl< const _Tp >` Class Template Reference

### Public Types

- `typedef const _Tp element_type`

### Public Member Functions

- `const _Tp * get () const`
- `bool operator< (const \_Relative\_pointer\_impl &__rarg) const`
- `bool operator== (const \_Relative\_pointer\_impl &__rarg) const`
- `void set (const _Tp *__arg)`

#### 4.25.1 Detailed Description

```
template<typename _Tp>class __gnu_cxx::_Relative_pointer_impl< const _Tp >
```

`Relative_pointer_impl` needs a specialization for `const T` because of the casting done during pointer arithmetic.

Definition at line 161 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

4.26 `__gnu_cxx::Std_pointer_impl<_Tp>` Class Template Reference

## Public Types

- `typedef _Tp element_type`

## Public Member Functions

- `_Tp * get () const`
- `bool operator< (const \_Std\_pointer\_impl &__rarg) const`
- `bool operator== (const \_Std\_pointer\_impl &__rarg) const`
- `void set (element_type *__arg)`

## 4.26.1 Detailed Description

```
template<typename _Tp>class __gnu_cxx::Std_pointer_impl<_Tp>
```

A storage policy for use with `_Pointer_adapter<>` which yields a standard pointer.

A `_Storage_policy` is required to provide 4 things: 1) A `get()` API for returning the stored pointer value. 2) An `set()` API for storing a pointer value. 3) An `element_type` typedef to define the type this points to. 4) An `operator<()` to support pointer comparison. 5) An `operator==()` to support pointer comparison.

Definition at line 66 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

4.27 `__gnu_cxx::Unqualified_type<_Tp>` Struct Template Reference

## Public Types

- `typedef _Tp type`

## 4.27.1 Detailed Description

```
template<typename _Tp>struct __gnu_cxx::Unqualified_type<_Tp>
```

This structure accommodates the way in which `std::iterator_traits<>` is normally specialized for `const T*`, so that `value_of(_type)` is still `T`.

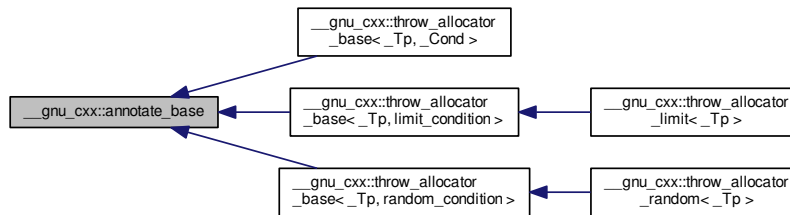
Definition at line 241 of file `pointer.h`.

The documentation for this struct was generated from the following file:

- [pointer.h](#)

## 4.28 `__gnu_cxx::annotate_base` Struct Reference

Inheritance diagram for `__gnu_cxx::annotate_base`:



### Public Member Functions

- void **check\_allocated** (void \*p, size\_t size)
- void **check\_allocated** (size\_t label)
- void **erase** (void \*p, size\_t size)
- void **insert** (void \*p, size\_t size)

### Static Public Member Functions

- static size\_t **get\_label** ()
- static void **set\_label** (size\_t l)

### Friends

- std::ostream & **operator**<< (std::ostream &, const [annotate\\_base](#) &)

#### 4.28.1 Detailed Description

Base class for checking address and label information about allocations. Create a `std::map` between the allocated address (void\*) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.

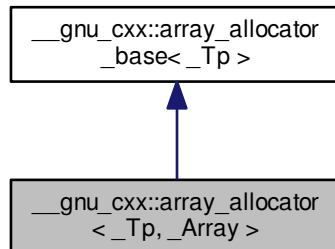
Definition at line 88 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

4.29 `__gnu_cxx::array_allocator<_Tp, _Array>` Class Template Reference

Inheritance diagram for `__gnu_cxx::array_allocator<_Tp, _Array>`:



## Public Types

- typedef `_Array` **array\_type**
- typedef const `_Tp *` **const\_pointer**
- typedef const `_Tp &` **const\_reference**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_Tp *` **pointer**
- typedef `std::true_type` **propagate\_on\_container\_move\_assignment**
- typedef `_Tp &` **reference**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- **array\_allocator** (`array_type *__array=0`) noexcept
- **array\_allocator** (const [array\\_allocator](#) &\_\_o) noexcept
- template<typename `_Tp1`, typename `_Array1`> **array\_allocator** (const [array\\_allocator](#)<`_Tp1`, `_Array1`> &) noexcept
- pointer **address** (reference `__x`) const noexcept
- const\_pointer **address** (const\_reference `__x`) const noexcept
- pointer **allocate** (`size_type __n`, const void \*=0)
- template<typename `_Up`, typename... `_Args`> void **construct** (`_Up *__p`, `_Args &&... __args`)
- void **deallocate** (pointer, `size_type`)
- template<typename `_Up`> void **destroy** (`_Up *__p`)
- `size_type` **max\_size** () const noexcept

## 4.29.1 Detailed Description

```
template<typename _Tp, typename _Array = std::tr1::array<_Tp, 1>> class __gnu_cxx::array_allocator<_Tp, _Array>
```

An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.



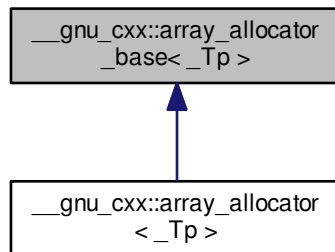
Definition at line 60 of file ext/alloc\_traits.h.

The documentation for this class was generated from the following files:

- [ext/alloc\\_traits.h](#)
- [array\\_allocator.h](#)

#### 4.30 \_\_gnu\_cxx::array\_allocator\_base< \_Tp > Class Template Reference

Inheritance diagram for \_\_gnu\_cxx::array\_allocator\_base< \_Tp >:



##### Public Types

- typedef const \_Tp \* **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef \_Tp \* **pointer**
- typedef \_Tp & **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

##### Public Member Functions

- pointer **address** (reference \_\_x) const noexcept
- const\_pointer **address** (const\_reference \_\_x) const noexcept
- template<typename \_Up, typename... \_Args> void **construct** (\_Up \*\_\_p, \_Args &&... \_\_args)
- void **deallocate** (pointer, size\_type)
- template<typename \_Up> void **destroy** (\_Up \*\_\_p)
- size\_type **max\_size** () const noexcept

##### 4.30.1 Detailed Description

```
template<typename _Tp>class __gnu_cxx::array_allocator_base< _Tp >
```

Base class.

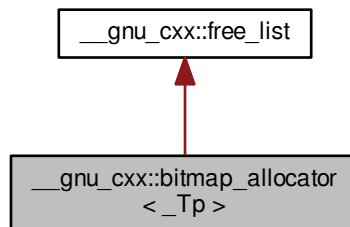
Definition at line 50 of file `array_allocator.h`.

The documentation for this class was generated from the following file:

- [array\\_allocator.h](#)

#### 4.31 `__gnu_cxx::bitmap_allocator<_Tp>` Class Template Reference

Inheritance diagram for `__gnu_cxx::bitmap_allocator<_Tp>`:



##### Public Types

- typedef `free_list::__mutex_type` **\_\_mutex\_type**
- typedef `const _Tp *` **const\_pointer**
- typedef `const _Tp &` **const\_reference**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_Tp *` **pointer**
- typedef `std::true_type` **propagate\_on\_container\_move\_assignment**
- typedef `_Tp &` **reference**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

##### Public Member Functions

- **bitmap\_allocator** (const [bitmap\\_allocator](#) &) noexcept
- template<typename `_Tp1` > **bitmap\_allocator** (const [bitmap\\_allocator](#)< `_Tp1` > &) noexcept
- pointer [\\_M\\_allocate\\_single\\_object](#) () throw (std::bad\_alloc)
- void [\\_M\\_deallocate\\_single\\_object](#) (pointer `__p`) throw ()
- pointer **address** (reference `__r`) const noexcept
- const\_pointer **address** (const\_reference `__r`) const noexcept
- pointer **allocate** (size\_type `__n`)
- pointer **allocate** (size\_type `__n`, typename [bitmap\\_allocator](#)< void >::const\_pointer)
- template<typename `_Up` , typename... `_Args`> void **construct** (`_Up` \* `__p`, `_Args` &&... `__args`)
- void **deallocate** (pointer `__p`, size\_type `__n`) throw ()
- template<typename `_Up` > void **destroy** (`_Up` \* `__p`)
- size\_type **max\_size** () const noexcept

## Private Types

- typedef vector\_type::iterator **iterator**
- typedef [\\_\\_detail::\\_\\_mini\\_vector](#)< value\_type > **vector\_type**

## Private Member Functions

- void [\\_M\\_clear](#) ()
- size\_t \* [\\_M\\_get](#) (size\_t \_\_sz) throw (std::bad\_alloc)
- void [\\_M\\_insert](#) (size\_t \*\_\_addr) throw ()

### 4.31.1 Detailed Description

template<typename \_Tp>class [\\_\\_gnu\\_cxx::bitmap\\_allocator](#)< \_Tp >

Bitmap Allocator, primary template.

Definition at line 70 of file ext/alloc\_traits.h.

### 4.31.2 Member Function Documentation

4.31.2.1 template<typename \_Tp > pointer [\\_\\_gnu\\_cxx::bitmap\\_allocator](#)< \_Tp >::\_M\_allocate\_single\_object ( ) throw std::bad\_alloc) [[inline](#)]

Allocates memory for a single object of size sizeof(\_Tp).

#### Exceptions

<a href="#">std::bad_alloc</a> .	If memory can not be allocated.
----------------------------------	---------------------------------

Complexity: Worst case complexity is O(N), but that is hardly ever hit. If and when this particular case is encountered, the next few cases are guaranteed to have a worst case complexity of O(1)! That's why this function performs very well on average. You can consider this function to have a complexity referred to commonly as: Amortized Constant time.

Definition at line 827 of file bitmap\_allocator.h.

References [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_bit\\_allocate\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_num\\_bitmaps\(\)](#), and [\\_\\_gnu\\_cxx::Bit\\_scan](#)↔[\\_forward\(\)](#).

4.31.2.2 template<typename \_Tp > void [\\_\\_gnu\\_cxx::bitmap\\_allocator](#)< \_Tp >::\_M\_deallocate\_single\_object ( pointer \_\_p ) throw ) [[inline](#)]

Deallocates memory that belongs to a single object of size sizeof(\_Tp).

Complexity: O(lg(N)), but the worst case is not hit often! This is because containers usually deallocate memory close to each other and this case is handled in O(1) time by the deallocate function.

Definition at line 917 of file bitmap\_allocator.h.

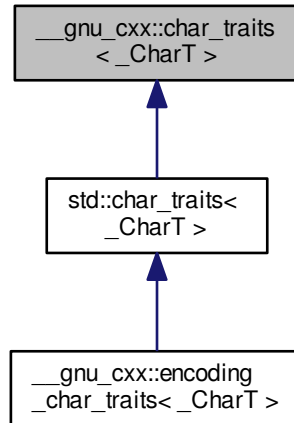
References [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_bit\\_free\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_num\\_bitmaps\(\)](#), [std::\\_\\_rotate\(\)](#), and [\\_\\_gnu\\_cxx::free\\_list::\\_M\\_insert\(\)](#).

The documentation for this class was generated from the following files:

- [ext/alloc\\_traits.h](#)
- [bitmap\\_allocator.h](#)

4.32 `__gnu_cxx::char_traits<_CharT>` Struct Template Reference

Inheritance diagram for `__gnu_cxx::char_traits<_CharT>`:



## Public Types

- typedef `_CharT char_type`
- typedef `_Char_types<_CharT>::int_type int_type`
- typedef `_Char_types<_CharT>::off_type off_type`
- typedef `_Char_types<_CharT>::pos_type pos_type`
- typedef `_Char_types<_CharT>::state_type state_type`

## Static Public Member Functions

- static void **assign** (`char_type &__c1`, `const char_type &__c2`)
- static `char_type *` **assign** (`char_type *__s`, `std::size_t __n`, `char_type __a`)
- static int **compare** (`const char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static `char_type *` **copy** (`char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static constexpr `int_type` **eof** ()
- static constexpr bool **eq** (`const char_type &__c1`, `const char_type &__c2`)
- static constexpr bool **eq\_int\_type** (`const int_type &__c1`, `const int_type &__c2`)
- static `const char_type *` **find** (`const char_type *__s`, `std::size_t __n`, `const char_type &__a`)
- static `std::size_t` **length** (`const char_type *__s`)
- static constexpr bool **lt** (`const char_type &__c1`, `const char_type &__c2`)
- static `char_type *` **move** (`char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static constexpr `int_type` **not\_eof** (`const int_type &__c`)
- static constexpr `char_type` **to\_char\_type** (`const int_type &__c`)
- static constexpr `int_type` **to\_int\_type** (`const char_type &__c`)

#### 4.32.1 Detailed Description

```
template<typename _CharT>struct __gnu_cxx::char_traits< _CharT >
```

Base class used to implement `std::char_traits`.

##### Note

For any given actual character type, this definition is probably wrong. (Most of the member functions are likely to be right, but the `int_type` and `state_type` typedefs, and the `eof()` member function, are likely to be wrong.) The reason this class exists is so users can specialize it. Classes in namespace `std` may not be specialized for fundamental types, but classes in namespace `__gnu_cxx` may be.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt05ch13s03.html> for advice on how to make use of this class for *unusual* character types. Also, check out `include/ext/pod_char_traits.h`.

Definition at line 83 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

### 4.33 \_\_gnu\_cxx::character< V, I, S > Struct Template Reference

#### Public Types

- typedef [character](#)< V, I, S > **char\_type**
- typedef I **int\_type**
- typedef S **state\_type**
- typedef V **value\_type**

#### Static Public Member Functions

- template<typename V2 > static [char\\_type](#) **from** (const V2 &v)
- template<typename V2 > static V2 **to** (const [char\\_type](#) &c)

#### Public Attributes

- `value_type` **value**

#### 4.33.1 Detailed Description

```
template<typename V, typename I, typename S = std::mbstate_t>struct __gnu_cxx::character< V, I, S >
```

A POD class that serves as a character abstraction class.

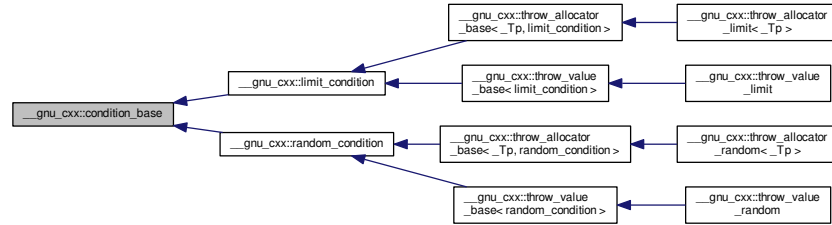
Definition at line 49 of file `pod_char_traits.h`.

The documentation for this struct was generated from the following file:

- [pod\\_char\\_traits.h](#)

## 4.34 \_\_gnu\_cxx::condition\_base Struct Reference

Inheritance diagram for \_\_gnu\_cxx::condition\_base:



## 4.34.1 Detailed Description

Base struct for condition policy.

Requires a public member function with the signature `void throw_conditionally()`

Definition at line 247 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 4.35 \_\_gnu\_cxx::debug\_allocator&lt;\_Alloc&gt; Class Template Reference

## Public Types

- typedef `_Alloc::const_pointer` **const\_pointer**
- typedef `_Alloc::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `_Alloc::pointer` **pointer**
- typedef `_Alloc::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `_Alloc::value_type` **value\_type**

## Public Member Functions

- pointer **allocate** (size\_type \_\_n)
- pointer **allocate** (size\_type \_\_n, const void \*\_\_hint)
- void **deallocate** (pointer \_\_p, size\_type \_\_n)

## 4.35.1 Detailed Description

```
template<typename _Alloc> class __gnu_cxx::debug_allocator<_Alloc>
```

A meta-allocator with debugging bits, as per [20.4].

This is precisely the allocator defined in the C++ Standard.

- all allocation calls operator new
- all deallocation calls operator delete

Definition at line 62 of file debug\_allocator.h.

The documentation for this class was generated from the following file:

- [debug\\_allocator.h](#)

#### 4.36 \_\_gnu\_cxx::enc\_filebuf<\_CharT> Class Template Reference

Inherits basic\_filebuf<\_CharT, encoding\_char\_traits<\_CharT>>.

##### Public Types

- typedef [traits\\_type::pos\\_type](#) **pos\_type**
- typedef [traits\\_type::state\\_type](#) **state\_type**
- typedef [encoding\\_char\\_traits<\\_CharT>](#) **traits\_type**

##### Public Member Functions

- **enc\_filebuf** ([state\\_type](#) &\_\_state)

##### 4.36.1 Detailed Description

```
template<typename _CharT>class __gnu_cxx::enc_filebuf<_CharT>
```

```
class enc_filebuf.
```

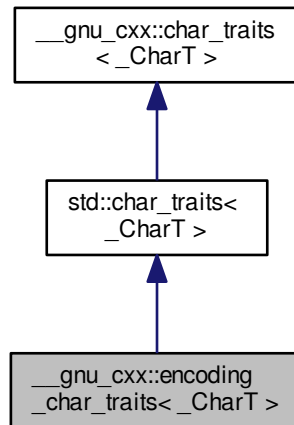
Definition at line 42 of file enc\_filebuf.h.

The documentation for this class was generated from the following file:

- [enc\\_filebuf.h](#)

4.37 `__gnu_cxx::encoding_char_traits<_CharT>` Struct Template Reference

Inheritance diagram for `__gnu_cxx::encoding_char_traits<_CharT>`:



## Public Types

- typedef `_CharT char_type`
- typedef `_Char_types<_CharT>::int_type int_type`
- typedef `_Char_types<_CharT>::off_type off_type`
- typedef `std::fpos<state_type> pos_type`
- typedef `encoding_state state_type`

## Static Public Member Functions

- static void **assign** (`char_type &__c1`, `const char_type &__c2`)
- static `char_type *` **assign** (`char_type *__s`, `std::size_t __n`, `char_type __a`)
- static int **compare** (`const char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static `char_type *` **copy** (`char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static constexpr `int_type` **eof** ()
- static constexpr bool **eq** (`const char_type &__c1`, `const char_type &__c2`)
- static constexpr bool **eq\_int\_type** (`const int_type &__c1`, `const int_type &__c2`)
- static `const char_type *` **find** (`const char_type *__s`, `std::size_t __n`, `const char_type &__a`)
- static `std::size_t` **length** (`const char_type *__s`)
- static constexpr bool **lt** (`const char_type &__c1`, `const char_type &__c2`)
- static `char_type *` **move** (`char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static constexpr `int_type` **not\_eof** (`const int_type &__c`)
- static constexpr `char_type` **to\_char\_type** (`const int_type &__c`)
- static constexpr `int_type` **to\_int\_type** (`const char_type &__c`)



#### 4.37.1 Detailed Description

template<typename \_CharT>struct \_\_gnu\_cxx::encoding\_char\_traits< \_CharT >

encoding\_char\_traits

Definition at line 210 of file codecvt\_specializations.h.

The documentation for this struct was generated from the following file:

- [codecvt\\_specializations.h](#)

### 4.38 \_\_gnu\_cxx::encoding\_state Class Reference

#### Public Types

- typedef iconv\_t **descriptor\_type**

#### Public Member Functions

- **encoding\_state** (const char \*\_\_int, const char \*\_\_ext, int \_\_ibom=0, int \_\_ebom=0, int \_\_bytes=1)
- **encoding\_state** (const [encoding\\_state](#) &\_\_obj)
- int **character\_ratio** () const
- int **external\_bom** () const
- const [std::string](#) **external\_encoding** () const
- bool **good** () const throw ()
- const descriptor\_type & **in\_descriptor** () const
- int **internal\_bom** () const
- const [std::string](#) **internal\_encoding** () const
- [encoding\\_state](#) & **operator=** (const [encoding\\_state](#) &\_\_obj)
- const descriptor\_type & **out\_descriptor** () const

#### Protected Member Functions

- void **construct** (const [encoding\\_state](#) &\_\_obj)
- void **destroy** () throw ()
- void **init** ()

#### Protected Attributes

- int **\_M\_bytes**
- int **\_M\_ext\_bom**
- [std::string](#) **\_M\_ext\_enc**
- descriptor\_type **\_M\_in\_desc**
- int **\_M\_int\_bom**
- [std::string](#) **\_M\_int\_enc**
- descriptor\_type **\_M\_out\_desc**

## 4.38.1 Detailed Description

Extension to use `iconv` for dealing with character encodings.

Definition at line 50 of file `codecvt_specializations.h`.

The documentation for this class was generated from the following file:

- [codecvt\\_specializations.h](#)

4.39 `__gnu_cxx::forced_error` Struct Reference

Inherits exception.

## 4.39.1 Detailed Description

Thrown by exception safety machinery.

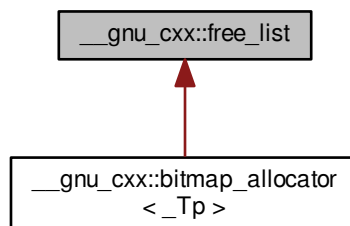
Definition at line 74 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

4.40 `__gnu_cxx::free_list` Class Reference

Inheritance diagram for `__gnu_cxx::free_list`:



## Public Types

- typedef `__mutex` **`mutex_type`**
- typedef `vector_type::iterator` **`iterator`**
- typedef `size_t * value_type`
- typedef `__detail::__mini_vector< value_type >` **`vector_type`**

## Public Member Functions

- void [\\_M\\_clear](#) ()
- size\_t \* [\\_M\\_get](#) (size\_t \_\_sz) throw (std::bad\_alloc)
- void [\\_M\\_insert](#) (size\_t \* \_\_addr) throw ()

### 4.40.1 Detailed Description

The free list class for managing chunks of memory to be given to and returned by the `bitmap_allocator`.  
Definition at line 521 of file `bitmap_allocator.h`.

### 4.40.2 Member Function Documentation

#### 4.40.2.1 void \_\_gnu\_cxx::free\_list::\_M\_clear ( )

This function just clears the internal Free List, and gives back all the memory to the OS.

#### 4.40.2.2 size\_t\* \_\_gnu\_cxx::free\_list::\_M\_get ( size\_t \_\_sz ) throw std::bad\_alloc

This function gets a block of memory of the specified size from the free list.

##### Parameters

<code>__sz</code>	The size in bytes of the memory required.
-------------------	---

##### Returns

A pointer to the new memory block of size at least equal to that requested.

#### 4.40.2.3 void \_\_gnu\_cxx::free\_list::\_M\_insert ( size\_t \* \_\_addr ) throw ) [inline]

This function returns the block of memory to the internal free list.

##### Parameters

<code>__addr</code>	The pointer to the memory block that was given by a call to the <code>_M_get</code> function.
---------------------	---

Definition at line 631 of file `bitmap_allocator.h`.

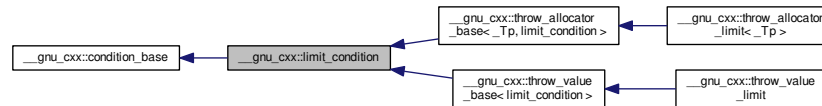
Referenced by `__gnu_cxx::bitmap_allocator<_Tp>::_M_deallocate_single_object()`.

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

## 4.41 \_\_gnu\_cxx::limit\_condition Struct Reference

Inheritance diagram for \_\_gnu\_cxx::limit\_condition:



## Classes

- struct [always\\_adjustor](#)
- struct [limit\\_adjustor](#)
- struct [never\\_adjustor](#)

## Static Public Member Functions

- static size\_t & **count** ()
- static size\_t & **limit** ()
- static void **set\_limit** (const size\_t \_\_l)
- static void **throw\_conditionally** ()

## 4.41.1 Detailed Description

Base class for incremental control and throw.

Definition at line 256 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 4.42 \_\_gnu\_cxx::limit\_condition::always\_adjustor Struct Reference

Inherits \_\_gnu\_cxx::limit\_condition::adjustor\_base.

## 4.42.1 Detailed Description

Always enter the condition.

Definition at line 280 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

#### 4.43 `__gnu_cxx::limit_condition::limit_adjustor` Struct Reference

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

##### Public Member Functions

- **`limit_adjustor`** (`const size_t __l`)

##### 4.43.1 Detailed Description

Enter the *n*th condition.

Definition at line 286 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

#### 4.44 `__gnu_cxx::limit_condition::never_adjustor` Struct Reference

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

##### 4.44.1 Detailed Description

Never enter the condition.

Definition at line 274 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

#### 4.45 `__gnu_cxx::malloc_allocator<_Tp>` Class Template Reference

##### Public Types

- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef std::true_type propagate_on_container_move_assignment`
- `typedef _Tp & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

##### Public Member Functions

- **`malloc_allocator`** (`const malloc\_allocator &`) `noexcept`
- `template<typename _Tp1 > malloc_allocator` (`const malloc\_allocator<_Tp1> &`) `noexcept`
- `pointer address` (`reference __x`) `const noexcept`
- `const_pointer address` (`const_reference __x`) `const noexcept`

- pointer **allocate** (size\_type \_\_n, const void \*=0)
- template<typename \_Up, typename... \_Args> void **construct** (\_Up \*\_\_p, \_Args &&...\_\_args)
- void **deallocate** (pointer \_\_p, size\_type)
- template<typename \_Up > void **destroy** (\_Up \*\_\_p)
- size\_type **max\_size** () const noexcept

#### 4.45.1 Detailed Description

template<typename \_Tp>class `__gnu_cxx::malloc_allocator<_Tp>`

An allocator that uses malloc.

This is precisely the allocator defined in the C++ Standard.

- all allocation calls malloc
- all deallocation calls free

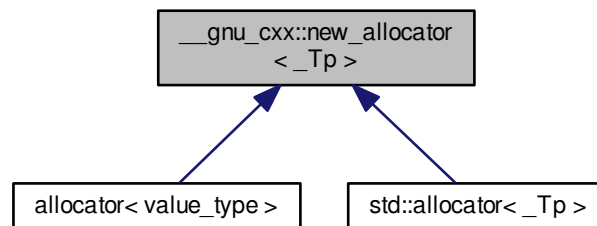
Definition at line 79 of file `ext/alloc_traits.h`.

The documentation for this class was generated from the following files:

- [ext/alloc\\_traits.h](#)
- [malloc\\_allocator.h](#)

## 4.46 `__gnu_cxx::new_allocator<_Tp>` Class Template Reference

Inheritance diagram for `__gnu_cxx::new_allocator<_Tp>`:



#### Public Types

- typedef const \_Tp \* **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef \_Tp \* **pointer**
- typedef std::true\_type **propagate\_on\_container\_move\_assignment**

- typedef `_Tp` & **reference**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

#### Public Member Functions

- **new\_allocator** (const [new\\_allocator](#) &) noexcept
- template<typename `_Tp1` > **new\_allocator** (const [new\\_allocator](#)< `_Tp1` > &) noexcept
- pointer **address** (reference `__x`) const noexcept
- const\_pointer **address** (const\_reference `__x`) const noexcept
- pointer **allocate** (size\_type `__n`, const void `*=0`)
- template<typename `_Up`, typename... `_Args`> void **construct** (`_Up` `*__p`, `_Args` &&...`__args`)
- void **deallocate** (pointer `__p`, size\_type)
- template<typename `_Up` > void **destroy** (`_Up` `*__p`)
- size\_type **max\_size** () const noexcept

#### 4.46.1 Detailed Description

template<typename `_Tp`>class `__gnu_cxx::new_allocator< _Tp >`

An allocator that uses global new, as per [20.4].

This is precisely the allocator defined in the C++ Standard.

- all allocation calls operator new
- all deallocation calls operator delete

#### Template Parameters

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

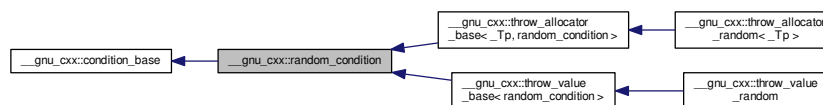
Definition at line 97 of file `ext/alloc_traits.h`.

The documentation for this class was generated from the following files:

- [ext/alloc\\_traits.h](#)
- [new\\_allocator.h](#)

#### 4.47 `__gnu_cxx::random_condition` Struct Reference

Inheritance diagram for `__gnu_cxx::random_condition`:



## Classes

- struct [always\\_adjustor](#)
- struct [group\\_adjustor](#)
- struct [never\\_adjustor](#)

## Public Member Functions

- void **seed** (unsigned long \_\_s)

## Static Public Member Functions

- static void **set\_probability** (double \_\_p)
- static void **throw\_conditionally** ()

## 4.47.1 Detailed Description

Base class for random probability control and throw.

Definition at line 328 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

4.48 `__gnu_cxx::random_condition::always_adjustor` Struct Reference

Inherits `__gnu_cxx::random_condition::adjustor_base`.

## 4.48.1 Detailed Description

Always enter the condition.

Definition at line 361 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

4.49 `__gnu_cxx::random_condition::group_adjustor` Struct Reference

Inherits `__gnu_cxx::random_condition::adjustor_base`.

## Public Member Functions

- **group\_adjustor** (size\_t size)



#### 4.49.1 Detailed Description

Group condition.

Definition at line 346 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

### 4.50 `__gnu_cxx::random_condition::never_adjustor` Struct Reference

Inherits `__gnu_cxx::random_condition::adjustor_base`.

#### 4.50.1 Detailed Description

Never enter the condition.

Definition at line 355 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

### 4.51 `__gnu_cxx::recursive_init_error` Class Reference

Inherits exception.

#### 4.51.1 Detailed Description

Exception thrown by `__cxa_guard_acquire`.

6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined.

Since we already have a library function to handle locking, we might as well check for this situation and throw an exception. We use the second byte of the guard variable to remember that we're in the middle of an initialization.

Definition at line 689 of file `cxxabi.h`.

The documentation for this class was generated from the following file:

- [cxxabi.h](#)

### 4.52 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference

Inherits `basic_filebuf<_CharT, _Traits>`.

#### Public Types

- typedef `_CharT` **char\_type**
- typedef `traits_type::int_type` **int\_type**
- typedef `traits_type::off_type` **off\_type**

- typedef traits\_type::pos\_type **pos\_type**
- typedef std::size\_t **size\_t**
- typedef \_Traits **traits\_type**

#### Public Member Functions

- `stdio_filebuf()`
- `stdio_filebuf(int __fd, std::ios_base::openmode __mode, size_t __size=static_cast<size_t>(BUFSIZ))`
- `stdio_filebuf(std::__c_file * __f, std::ios_base::openmode __mode, size_t __size=static_cast<size_t>(BUFSIZ))`
- virtual `~stdio_filebuf()`
- `int fd()`
- `std::__c_file * file()`

#### 4.52.1 Detailed Description

template<typename \_CharT, typename \_Traits = std::char\_traits<\_CharT>> class `__gnu_cxx::stdio_filebuf<_CharT, _Traits>`

Provides a layer of compatibility for C/POSIX.

This GNU extension provides extensions for working with standard C FILE\*'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

Definition at line 50 of file `stdio_filebuf.h`.

#### 4.52.2 Constructor & Destructor Documentation

4.52.2.1 template<typename \_CharT, typename \_Traits = std::char\_traits<\_CharT>> `__gnu_cxx::stdio_filebuf<_CharT, _Traits>::stdio_filebuf( )` [inline]

deferred initialization

Definition at line 65 of file `stdio_filebuf.h`.

4.52.2.2 template<typename \_CharT, typename \_Traits> `__gnu_cxx::stdio_filebuf<_CharT, _Traits>::stdio_filebuf( int __fd, std::ios_base::openmode __mode, size_t __size = static_cast<size_t>(BUFSIZ) )`

##### Parameters

<code>__fd</code>	An open file descriptor.
<code>__mode</code>	Same meaning as in a standard filebuf.
<code>__size</code>	Optimal or preferred size of internal buffer, in chars.

This constructor associates a file stream buffer with an open POSIX file descriptor. The file descriptor will be automatically closed when the `stdio_filebuf` is closed/destroyed.

Definition at line 128 of file `stdio_filebuf.h`.

4.52.2.3 template<typename \_CharT, typename \_Traits> `__gnu_cxx::stdio_filebuf<_CharT, _Traits>::stdio_filebuf( std::__c_file * __f, std::ios_base::openmode __mode, size_t __size = static_cast<size_t>(BUFSIZ) )`

## Parameters

<code>__f</code>	An open <code>FILE*</code> .
<code>__mode</code>	Same meaning as in a standard filebuf.
<code>__size</code>	Optimal or preferred size of internal buffer, in chars. Defaults to system's <code>BUFSIZ</code> .

This constructor associates a file stream buffer with an open C `FILE*`. The `FILE*` will not be automatically closed when the `stdio_filebuf` is closed/destroyed.

Definition at line 144 of file `stdio_filebuf.h`.

4.52.2.4 `template<typename _CharT, typename _Traits > __gnu_cxx::stdio_filebuf< _CharT, _Traits >::~~stdio_filebuf ( )`  
[virtual]

Closes the external data stream if the file descriptor constructor was used.

Definition at line 123 of file `stdio_filebuf.h`.

## 4.52.3 Member Function Documentation

4.52.3.1 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> int __gnu_cxx::stdio_filebuf< _CharT, _Traits >::fd ( )` [inline]

## Returns

The underlying file descriptor.

Once associated with an external data stream, this function can be used to access the underlying POSIX file descriptor. Note that there is no way for the library to track what you do with the descriptor, so be careful.

Definition at line 109 of file `stdio_filebuf.h`.

4.52.3.2 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> std::_c_file* __gnu_cxx::stdio_filebuf< _CharT, _Traits >::file ( )` [inline]

## Returns

The underlying `FILE*`.

This function can be used to access the underlying "C" file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

Definition at line 119 of file `stdio_filebuf.h`.

The documentation for this class was generated from the following file:

- [stdio\\_filebuf.h](#)

4.53 `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >` Class Template Reference

Inherits `basic_streambuf< _CharT, _Traits >`.

## Public Types

- typedef `_CharT` **char\_type**
- typedef `traits_type::int_type` **int\_type**
- typedef `traits_type::off_type` **off\_type**

- typedef traits\_type::pos\_type **pos\_type**
- typedef \_Traits **traits\_type**

#### Public Member Functions

- **stdio\_sync\_filebuf** (std::\_\_c\_file \* \_\_f)
- std::\_\_c\_file \*const **file** ()

#### Protected Member Functions

- virtual int\_type **overflow** (int\_type \_\_c=traits\_type::eof())
- virtual int\_type **pbackfail** (int\_type \_\_c=traits\_type::eof())
- virtual [std::streampos seekoff](#) ([std::streamoff](#) \_\_off, [std::ios\\_base::seekdir](#) \_\_dir, [std::ios\\_base::openmode](#)=[std::ios\\_base::in](#)|[std::ios\\_base::out](#))
- virtual [std::streampos seekpos](#) ([std::streampos](#) \_\_pos, [std::ios\\_base::openmode](#) \_\_mode=[std::ios\\_base::in](#)|[std::ios\\_base::out](#))
- virtual int **sync** ()
- int\_type **syncgetc** ()
- template<> [stdio\\_sync\\_filebuf<char>::int\\_type syncgetc](#) ()
- template<> [stdio\\_sync\\_filebuf<wchar\\_t>::int\\_type syncgetc](#) ()
- int\_type **syncputc** (int\_type \_\_c)
- template<> [stdio\\_sync\\_filebuf<char>::int\\_type syncputc](#) (int\_type \_\_c)
- template<> [stdio\\_sync\\_filebuf<wchar\\_t>::int\\_type syncputc](#) (int\_type \_\_c)
- int\_type **syncungetc** (int\_type \_\_c)
- template<> [stdio\\_sync\\_filebuf<char>::int\\_type syncungetc](#) (int\_type \_\_c)
- template<> [stdio\\_sync\\_filebuf<wchar\\_t>::int\\_type syncungetc](#) (int\_type \_\_c)
- virtual int\_type **uflow** ()
- virtual int\_type **underflow** ()
- virtual [std::streamsize xsgetn](#) (char\_type \* \_\_s, [std::streamsize](#) \_\_n)
- template<> [std::streamsize xsgetn](#) (char \* \_\_s, [std::streamsize](#) \_\_n)
- template<> [std::streamsize xsgetn](#) (wchar\_t \* \_\_s, [std::streamsize](#) \_\_n)
- virtual [std::streamsize xspun](#) (const char\_type \* \_\_s, [std::streamsize](#) \_\_n)
- template<> [std::streamsize xspun](#) (const char \* \_\_s, [std::streamsize](#) \_\_n)
- template<> [std::streamsize xspun](#) (const wchar\_t \* \_\_s, [std::streamsize](#) \_\_n)

#### 4.53.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>> class __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>
```

Provides a layer of compatibility for C.

This GNU extension provides extensions for working with standard C FILE\*'s. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

Definition at line 56 of file `stdio_sync_filebuf.h`.

#### 4.53.2 Member Function Documentation

4.53.2.1 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> std::_c_file* const  
__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::file ( ) [inline]`

##### Returns

The underlying FILE\*.

This function can be used to access the underlying C file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

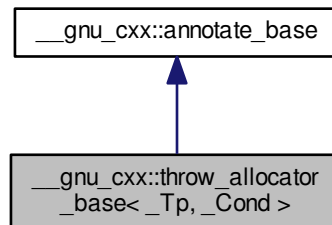
Definition at line 88 of file `stdio_sync_filebuf.h`.

The documentation for this class was generated from the following file:

- [stdio\\_sync\\_filebuf.h](#)

#### 4.54 `__gnu_cxx::throw_allocator_base<_Tp, _Cond>` Class Template Reference

Inheritance diagram for `__gnu_cxx::throw_allocator_base<_Tp, _Cond>`:



##### Public Types

- `typedef const value_type * const_pointer`
- `typedef const value_type & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef value_type * pointer`
- `typedef std::true_type propagate_on_container_move_assignment`
- `typedef value_type & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

##### Public Member Functions

- `pointer address (reference __x) const noexcept`
- `const_pointer address (const_reference __x) const noexcept`

- pointer **allocate** (size\_type \_\_n, [std::allocator](#)< void >::const\_pointer hint=0)
- void **check\_allocated** (void \*p, size\_t size)
- void **check\_allocated** (pointer \_\_p, size\_type \_\_n)
- void **check\_allocated** (size\_type \_\_n)
- template<typename \_Up, typename... \_Args> void **construct** (\_Up \*\_\_p, \_Args &&... \_\_args)
- void **deallocate** (pointer \_\_p, size\_type \_\_n)
- template<typename \_Up > void **destroy** (\_Up \*\_\_p)
- void **erase** (void \*p, size\_t size)
- void **insert** (void \*p, size\_t size)
- size\_type **max\_size** () const noexcept

#### Static Public Member Functions

- static size\_t **get\_label** ()
- static void **set\_label** (size\_t l)

##### 4.54.1 Detailed Description

template<typename \_Tp, typename \_Cond>class `__gnu_cxx::throw_allocator_base<_Tp, _Cond>`

Allocator class with logging and exception generation control. Intended to be used as an `allocator_type` in templated code.

Note: Deallocate not allowed to throw.

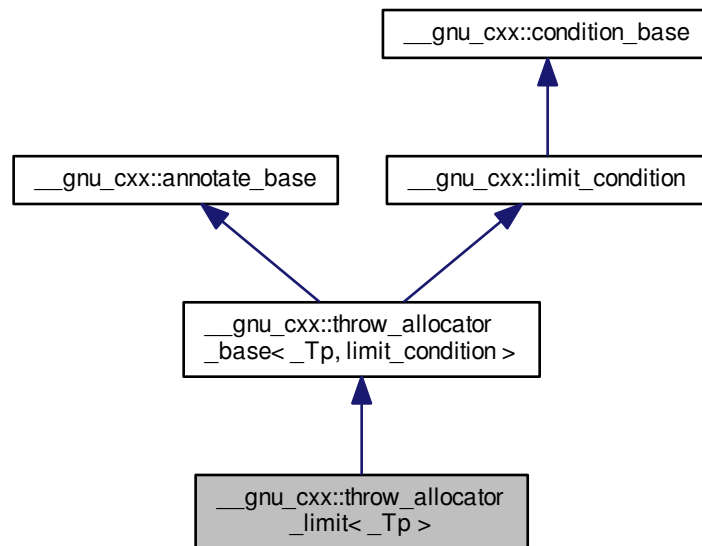
Definition at line 634 of file `throw_allocator.h`.

The documentation for this class was generated from the following file:

- [throw\\_allocator.h](#)

#### 4.55 `__gnu_cxx::throw_allocator_limit<_Tp>` Struct Template Reference

Inheritance diagram for `__gnu_cxx::throw_allocator_limit<_Tp>`:



##### Public Types

- typedef const value\_type \* **const\_pointer**
- typedef const value\_type & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef value\_type \* **pointer**
- typedef std::true\_type **propagate\_on\_container\_move\_assignment**
- typedef value\_type & **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

##### Public Member Functions

- **throw\_allocator\_limit** (const [throw\\_allocator\\_limit](#) &) noexcept
- template<typename \_Tp1 > **throw\_allocator\_limit** (const [throw\\_allocator\\_limit](#)<\_Tp1 > &) noexcept
- pointer **address** (reference \_\_x) const noexcept
- const\_pointer **address** (const\_reference \_\_x) const noexcept
- pointer **allocate** (size\_type \_\_n, [std::allocator](#)< void >::const\_pointer hint=0)
- void **check\_allocated** (void \*p, size\_t size)
- void **check\_allocated** (pointer \_\_p, size\_type \_\_n)
- void **check\_allocated** (size\_type \_\_n)
- void **construct** (\_Up \*\_\_p, \_Args &&... \_\_args)

- void **deallocate** (pointer \_\_p, size\_type \_\_n)
- void **destroy** (\_Up \*\_\_p)
- void **erase** (void \*p, size\_t size)
- void **insert** (void \*p, size\_t size)
- size\_type **max\_size** () const noexcept

#### Static Public Member Functions

- static size\_t & **count** ()
- static size\_t **get\_label** ()
- static size\_t & **limit** ()
- static void **set\_label** (size\_t l)
- static void **set\_limit** (const size\_t \_\_l)
- static void **throw\_conditionally** ()

#### 4.55.1 Detailed Description

`template<typename _Tp>struct __gnu_cxx::throw_allocator_limit<_Tp>`

Allocator throwing via limit condition.

Definition at line 737 of file `throw_allocator.h`.

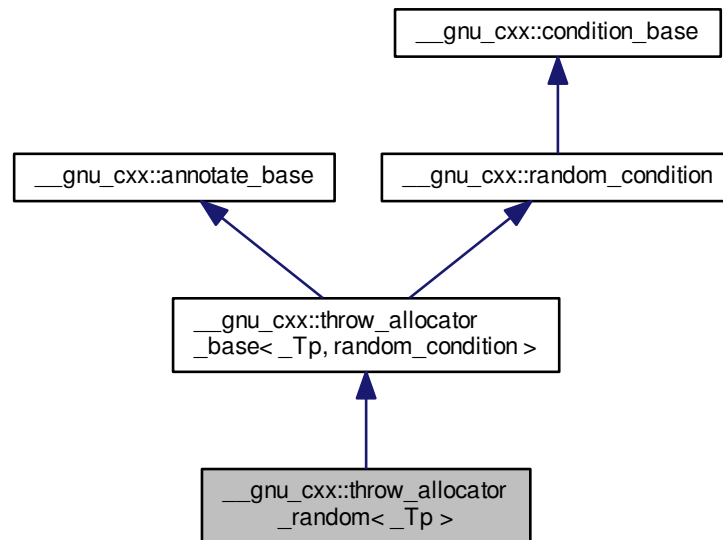
The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)



#### 4.56 `__gnu_cxx::throw_allocator_random<_Tp>` Struct Template Reference

Inheritance diagram for `__gnu_cxx::throw_allocator_random<_Tp>`:



##### Public Types

- typedef const value\_type \* **const\_pointer**
- typedef const value\_type & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef value\_type \* **pointer**
- typedef std::true\_type **propagate\_on\_container\_move\_assignment**
- typedef value\_type & **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

##### Public Member Functions

- **throw\_allocator\_random** (const [throw\\_allocator\\_random](#) &) noexcept
- template<typename \_Tp1 > **throw\_allocator\_random** (const [throw\\_allocator\\_random](#)<\_Tp1> &) noexcept
- pointer **address** (reference \_\_x) const noexcept
- const\_pointer **address** (const\_reference \_\_x) const noexcept
- pointer **allocate** (size\_type \_\_n, [std::allocator](#)< void >::const\_pointer hint=0)
- void **check\_allocated** (void \*p, size\_t size)
- void **check\_allocated** (pointer \_\_p, size\_type \_\_n)
- void **check\_allocated** (size\_type \_\_n)
- void **construct** (\_Up \*\_\_p, \_Args &&... \_\_args)

- void **deallocate** (pointer `__p`, size\_type `__n`)
- void **destroy** (`_Up *``__p`)
- void **erase** (void `*p`, size\_t `size`)
- void **insert** (void `*p`, size\_t `size`)
- size\_type **max\_size** () const noexcept
- void **seed** (unsigned long `__s`)

#### Static Public Member Functions

- static size\_t **get\_label** ()
- static void **set\_label** (size\_t `l`)
- static void **set\_probability** (double `__p`)
- static void **throw\_conditionally** ()

#### 4.56.1 Detailed Description

template<typename `_Tp`>struct `__gnu_cxx::throw_allocator_random<_Tp>`

Allocator throwing via random condition.

Definition at line 758 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 4.57 `__gnu_cxx::throw_value_base<_Cond>` Struct Template Reference

Inherits `_Cond`.

#### Public Types

- typedef `_Cond` **condition\_type**

#### Public Member Functions

- **throw\_value\_base** (const [throw\\_value\\_base](#) &`__v`)
- **throw\_value\_base** ([throw\\_value\\_base](#) &&)=default
- **throw\_value\_base** (const std::size\_t `__i`)
- [throw\\_value\\_base](#) & **operator++** ()
- [throw\\_value\\_base](#) & **operator=** (const [throw\\_value\\_base](#) &`__v`)
- [throw\\_value\\_base](#) & **operator=** ([throw\\_value\\_base](#) &&)=default

#### Public Attributes

- std::size\_t **M\_i**

#### 4.57.1 Detailed Description

```
template<typename _Cond>struct __gnu_cxx::throw_value_base< _Cond >
```

Class with exception generation control. Intended to be used as a value\_type in templized code.

Note: Destructor not allowed to throw.

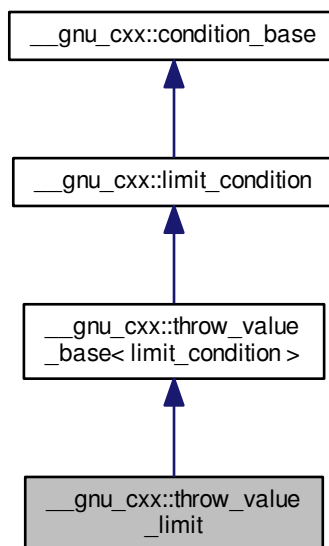
Definition at line 447 of file throw\_allocator.h.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

#### 4.58 \_\_gnu\_cxx::throw\_value\_limit Struct Reference

Inheritance diagram for \_\_gnu\_cxx::throw\_value\_limit:



#### Public Types

- typedef [throw\\_value\\_base< limit\\_condition >](#) **base\_type**
- typedef [limit\\_condition](#) **condition\_type**

#### Public Member Functions

- **throw\_value\_limit** (const [throw\\_value\\_limit](#) &\_\_other)
- **throw\_value\_limit** ([throw\\_value\\_limit](#) &&)=default

- `throw_value_limit` (const std::size\_t \_\_i)
- `throw_value_base` & `operator++` ()
- `throw_value_limit` & `operator=` (const `throw_value_limit` &\_\_other)
- `throw_value_limit` & `operator=` (`throw_value_limit` &&)=default

#### Static Public Member Functions

- static size\_t & `count` ()
- static size\_t & `limit` ()
- static void `set_limit` (const size\_t \_\_l)
- static void `throw_conditionally` ()

#### Public Attributes

- std::size\_t `_M_i`

##### 4.58.1 Detailed Description

Type throwing via limit condition.

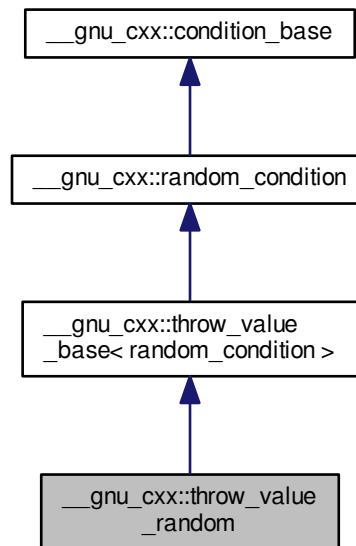
Definition at line 564 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- `throw_allocator.h`

#### 4.59 `__gnu_cxx::throw_value_random` Struct Reference

Inheritance diagram for `__gnu_cxx::throw_value_random`:



##### Public Types

- typedef `throw_value_base< random_condition >` `base_type`
- typedef `random_condition` `condition_type`

##### Public Member Functions

- `throw_value_random` (const `throw_value_random` &\_\_other)
- `throw_value_random` (`throw_value_random` &&)=default
- `throw_value_random` (const std::size\_t \_\_i)
- `throw_value_base` & `operator++` ()
- `throw_value_random` & `operator=` (const `throw_value_random` &\_\_other)
- `throw_value_random` & `operator=` (`throw_value_random` &&)=default
- void `seed` (unsigned long \_\_s)

##### Static Public Member Functions

- static void `set_probability` (double \_\_p)
- static void `throw_conditionally` ()

## Public Attributes

- `std::size_t _M_i`

## 4.59.1 Detailed Description

Type throwing via random condition.

Definition at line 595 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

4.60 `__gnu_debug::After_nth_from<_Iterator>` Class Template Reference

## Public Member Functions

- `_After_nth_from` (const difference\_type &\_\_n, const \_Iterator &\_\_base)
- `bool operator()` (const \_Iterator &\_\_x) const

## 4.60.1 Detailed Description

```
template<typename _Iterator>class __gnu_debug::After_nth_from<_Iterator>
```

A function object that returns true when the given random access iterator is at least `n` steps away from the given iterator.

Definition at line 77 of file `safe_sequence.h`.

The documentation for this class was generated from the following file:

- [safe\\_sequence.h](#)

4.61 `__gnu_debug::BeforeBeginHelper<_Sequence>` Struct Template Reference

## Public Types

- `typedef _It::iterator_type _Baselt`
- `typedef _Sequence::const_iterator _It`

## Static Public Member Functions

- `static bool _S_Is` (\_Baselt, const \_Sequence \*)
- `static bool _S_Is_Beginnest` (\_Baselt \_\_it, const \_Sequence \*\_\_seq)

## 4.61.1 Detailed Description

```
template<typename _Sequence>struct __gnu_debug::BeforeBeginHelper<_Sequence>
```

Helper struct to deal with sequence offering a `before_begin` iterator.

Definition at line 45 of file `safe_iterator.h`.

The documentation for this struct was generated from the following file:

- [safe\\_iterator.h](#)

## 4.62 `__gnu_debug::Equal_to<_Type>` Class Template Reference

### Public Member Functions

- `_Equal_to` (const `_Type` &\_\_v)
- bool `operator()` (const `_Type` &\_\_x) const

#### 4.62.1 Detailed Description

```
template<typename _Type>class __gnu_debug::Equal_to<_Type>
```

A simple function object that returns true if the passed-in value is equal to the stored value.

Definition at line 62 of file `safe_sequence.h`.

The documentation for this class was generated from the following file:

- [safe\\_sequence.h](#)

## 4.63 `__gnu_debug::Not_equal_to<_Type>` Class Template Reference

### Public Member Functions

- `_Not_equal_to` (const `_Type` &\_\_v)
- bool `operator()` (const `_Type` &\_\_x) const

#### 4.63.1 Detailed Description

```
template<typename _Type>class __gnu_debug::Not_equal_to<_Type>
```

A simple function object that returns true if the passed-in value is not equal to the stored value. It saves typing over using both `bind1st` and `not_equal`.

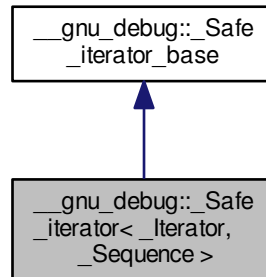
Definition at line 47 of file `safe_sequence.h`.

The documentation for this class was generated from the following file:

- [safe\\_sequence.h](#)

## 4.64 \_\_gnu\_debug::\_Safe\_iterator&lt;\_Iterator, \_Sequence&gt; Class Template Reference

Inheritance diagram for \_\_gnu\_debug::\_Safe\_iterator<\_Iterator, \_Sequence>:



## Public Types

- typedef \_Traits::difference\_type **difference\_type**
- typedef \_Traits::iterator\_category **iterator\_category**
- typedef \_Iterator **iterator\_type**
- typedef \_Traits::pointer **pointer**
- typedef \_Traits::reference **reference**
- typedef \_Traits::value\_type **value\_type**

## Public Member Functions

- [\\_Safe\\_iterator](#) ()
- [\\_Safe\\_iterator](#) (const \_Iterator &\_\_i, const \_Sequence \*\_\_seq)
- [\\_Safe\\_iterator](#) (const [\\_Safe\\_iterator](#) &\_\_x)
- [\\_Safe\\_iterator](#) ([\\_Safe\\_iterator](#) &&\_\_x)
- template<typename \_MutableIterator > [\\_Safe\\_iterator](#) (const [\\_Safe\\_iterator](#)<\_MutableIterator, typename \_\_gnu\_cxx::\_\_enable\_if<(std::\_\_are\_same<\_MutableIterator, typename \_Sequence::iterator::iterator\_type >::\_\_value), \_\_Sequence >::\_\_type > &\_\_x)
- void [\\_M\\_attach](#) ([\\_Safe\\_sequence\\_base](#) \*\_\_seq, bool \_\_constant)
- void [\\_M\\_attach](#) ([\\_Safe\\_sequence\\_base](#) \*\_\_seq)
- void [\\_M\\_attach\\_single](#) ([\\_Safe\\_sequence\\_base](#) \*\_\_seq, bool \_\_constant) throw ()
- void [\\_M\\_attach\\_single](#) ([\\_Safe\\_sequence\\_base](#) \*\_\_seq)
- bool [\\_M\\_attached\\_to](#) (const [\\_Safe\\_sequence\\_base](#) \*\_\_seq) const
- bool [\\_M\\_before\\_dereferenceable](#) () const
- bool [\\_M\\_can\\_advance](#) (const difference\_type &\_\_n) const
- bool [\\_M\\_can\\_compare](#) (const [\\_Safe\\_iterator\\_base](#) &\_\_x) const throw ()
- bool [\\_M\\_decrementable](#) () const
- bool [\\_M\\_dereferenceable](#) () const
- void [\\_M\\_detach](#) ()
- void [\\_M\\_detach\\_single](#) () throw ()



- `const _Sequence * _M_get_sequence () const`
- `bool _M_incrementable () const`
- `void _M_invalidate ()`
- `bool _M_is_before_begin () const`
- `bool _M_is_begin () const`
- `bool _M_is_beginnest () const`
- `bool _M_is_end () const`
- `void _M_reset () throw ()`
- `bool _M_singular () const throw ()`
- `void _M_unlink () throw ()`
- `template<typename _Other > bool _M_valid_range (const _Safe_iterator< _Other, _Sequence > &__rhs) const`
- `_Iterator base () const`
- `operator _Iterator () const`
- `reference operator* () const`
- `_Safe_iterator operator+ (const difference_type &__n) const`
- `_Safe_iterator & operator++ ()`
- `_Safe_iterator operator++ (int)`
- `_Safe_iterator & operator+= (const difference_type &__n)`
- `_Safe_iterator operator- (const difference_type &__n) const`
- `_Safe_iterator & operator-- ()`
- `_Safe_iterator operator-- (int)`
- `_Safe_iterator & operator-= (const difference_type &__n)`
- `pointer operator-> () const`
- `_Safe_iterator & operator= (const _Safe_iterator &__x)`
- `_Safe_iterator & operator= (_Safe_iterator &&__x)`
- `reference operator[] (const difference_type &__n) const`

#### Public Attributes

- `_Safe_iterator_base * _M_next`
- `_Safe_iterator_base * _M_prior`
- `_Safe_sequence_base * _M_sequence`
- `unsigned int _M_version`

#### Protected Member Functions

- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`

#### 4.64.1 Detailed Description

```
template<typename _Iterator, typename _Sequence>class __gnu_debug::_Safe_iterator< _Iterator, _Sequence >
```

Safe iterator wrapper.

The class template `_Safe_iterator` is a wrapper around an iterator that tracks the iterator's movement among sequences and checks that operations performed on the "safe" iterator are legal. In addition to the basic iterator operations (which are validated, and then passed to the underlying iterator), `_Safe_iterator` has member functions for iterator invalidation, attaching/detaching the iterator from sequences, and querying the iterator's state.

Definition at line 46 of file `formatter.h`.

## 4.64.2 Constructor &amp; Destructor Documentation

4.64.2.1 `template<typename _Iterator, typename _Sequence> __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_Safe_iterator ( ) [inline]`

## Postcondition

the iterator is singular and unattached

Definition at line 142 of file `safe_iterator.h`.

4.64.2.2 `template<typename _Iterator, typename _Sequence> __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_Safe_iterator ( const _Iterator & __i, const _Sequence * __seq ) [inline]`

Safe iterator construction from an unsafe iterator and its sequence.

## Precondition

`seq` is not NULL

## Postcondition

this is not singular

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

References `__gnu_debug::_Safe_iterator_base::_M_singular()`.

4.64.2.3 `template<typename _Iterator, typename _Sequence> __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_Safe_iterator ( const _Safe_iterator<_Iterator, _Sequence> & __x ) [inline]`

Copy construction.

Definition at line 162 of file `safe_iterator.h`.

4.64.2.4 `template<typename _Iterator, typename _Sequence> __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_Safe_iterator ( _Safe_iterator<_Iterator, _Sequence> && __x ) [inline]`

Move construction.

## Postcondition

`__x` is singular and unattached

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

References `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_attach()`, and `std::swap()`.

4.64.2.5 `template<typename _Iterator, typename _Sequence> template<typename _MutableIterator> __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_Safe_iterator ( const _Safe_iterator<_MutableIterator, typename __gnu_cxx::__enable_if<(std::__are_same<_MutableIterator, typename _Sequence::iterator::iterator_type>>::value), _Sequence>::_type > & __x ) [inline]`

Converting constructor from a mutable iterator to a constant iterator.

Definition at line 197 of file `safe_iterator.h`.

#### 4.64.3 Member Function Documentation

**4.64.3.1** `void __gnu_debug::Safe_iterator_base::M_attach ( _Safe_sequence_base * __seq, bool __constant )`  
[inherited]

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

Referenced by `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >::M_attach()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_attach()`, and `__gnu_debug::Safe_iterator_base::Safe_iterator_base()`.

**4.64.3.2** `template<typename _Iterator, typename _Sequence> void __gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_attach ( _Safe_sequence_base * __seq )` [inline]

Attach iterator to the given sequence.

Definition at line 406 of file `safe_iterator.h`.

References `__gnu_debug::Safe_iterator_base::M_attach()`.

Referenced by `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::Safe_iterator()`, and `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::operator=()`.

**4.64.3.3** `void __gnu_debug::Safe_iterator_base::M_attach_single ( _Safe_sequence_base * __seq, bool __constant ) throw`  
[inherited]

Likewise, but not thread-safe.

Referenced by `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >::M_attach_single()`, and `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_attach_single()`.

**4.64.3.4** `template<typename _Iterator, typename _Sequence> void __gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_attach_single ( _Safe_sequence_base * __seq )` [inline]

Likewise, but not thread-safe.

Definition at line 413 of file `safe_iterator.h`.

References `__gnu_debug::Safe_iterator_base::M_attach_single()`.

**4.64.3.5** `bool __gnu_debug::Safe_iterator_base::M_attached_to ( const _Safe_sequence_base * __seq ) const`  
[inline], [inherited]

Determines if we are attached to the given sequence.

Definition at line 129 of file `safe_base.h`.

**4.64.3.6** `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_before_dereferenceable ( ) const` [inline]

Is the iterator before a dereferenceable one?

Definition at line 425 of file `safe_iterator.h`.

References `__gnu_debug::base()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_incrementable()`, and `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::base()`.

4.64.3.7 `bool __gnu_debug::Safe_iterator_base::M_can_compare ( const _Safe_iterator_base & __x ) const throw ()`  
`[inherited]`

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

4.64.3.8 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_dereferenceable ( ) const` `[inline]`

Is the iterator dereferenceable?

Definition at line 420 of file `safe_iterator.h`.

References `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_is_before_begin()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_is_end()`, and `__gnu_debug::Safe_iterator_base::M_singular()`.

Referenced by `__gnu_debug::check_dereferenceable()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::operator*()`, and `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::operator->()`.

4.64.3.9 `void __gnu_debug::Safe_iterator_base::M_detach ( )` `[inherited]`

Detach the iterator for whatever sequence it is attached to, if any.

4.64.3.10 `void __gnu_debug::Safe_iterator_base::M_detach_single ( ) throw ()` `[inherited]`

Likewise, but not thread-safe.

4.64.3.11 `__gnu_cxx::mutex& __gnu_debug::Safe_iterator_base::M_get_mutex ( ) throw ()` `[protected]`,  
`[inherited]`

For use in `_Safe_iterator`.

4.64.3.12 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_incrementable ( ) const` `[inline]`

Is the iterator incrementable?

Definition at line 437 of file `safe_iterator.h`.

References `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_is_end()`, and `__gnu_debug::Safe_iterator_base::M_singular()`.

Referenced by `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_before_dereferenceable()`, and `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::operator++()`.

4.64.3.13 `void __gnu_debug::Safe_iterator_base::M_invalidate ( )` `[inline]`, `[inherited]`

Invalidate the iterator, making it singular.

Definition at line 142 of file `safe_base.h`.

4.64.3.14 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_is_before_begin ( ) const` `[inline]`

Is this iterator equal to the sequence's `before_begin()` iterator if any?

Definition at line 468 of file `safe_iterator.h`.

References `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::base()`.

Referenced by `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_dereferenceable()`.

4.64.3.15 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_iterator< _Iterator, _Sequence >::_M_is_begin ( ) const [inline]`

Is this iterator equal to the sequence's begin() iterator?

Definition at line 459 of file `safe_iterator.h`.

References `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::base()`.

4.64.3.16 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_iterator< _Iterator, _Sequence >::_M_is_beginnest ( ) const [inline]`

Is this iterator equal to the sequence's before\_begin() iterator if any or begin() otherwise?

Definition at line 475 of file `safe_iterator.h`.

References `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::base()`.

4.64.3.17 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_iterator< _Iterator, _Sequence >::_M_is_end ( ) const [inline]`

Is this iterator equal to the sequence's end() iterator?

Definition at line 463 of file `safe_iterator.h`.

References `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::base()`.

Referenced by `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable()`, and `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::_M_incrementable()`.

4.64.3.18 `void __gnu_debug::Safe_iterator_base::_M_reset ( ) throw ) [inherited]`

Reset all member variables

4.64.3.19 `bool __gnu_debug::Safe_iterator_base::_M_singular ( ) const throw ) [inherited]`

Is this iterator singular?

Referenced by `__gnu_debug::check_singular()`, `__gnu_debug::check_singular_aux()`, `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >::_M_dereferenceable()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable()`, `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >::_M_incrementable()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::_M_incrementable()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::Safe_iterator()`, and `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >::Safe_local_iterator()`.

4.64.3.20 `void __gnu_debug::Safe_iterator_base::_M_unlink ( ) throw ) [inline],[inherited]`

Unlink itself

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

References `__gnu_debug::Safe_iterator_base::_M_next`, and `__gnu_debug::Safe_iterator_base::_M_prior`.

4.64.3.21 `template<typename _Iterator, typename _Sequence> _Iterator __gnu_debug::Safe_iterator< _Iterator, _Sequence >::base ( ) const [inline]`

Return the underlying iterator.

Definition at line 396 of file `safe_iterator.h`.

Referenced by `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::_M_before_dereferenceable()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::_M_is_before_begin()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence`

`>::M_is_begin()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_is_beginnest()`, and `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_is_end()`.

4.64.3.22 `template<typename _Iterator, typename _Sequence> __gnu_debug::Safe_iterator<_Iterator, _Sequence>::operator _Iterator ( ) const [inline]`

Conversion to underlying non-debug iterator to allow better interaction with non-debug containers.

Definition at line 402 of file `safe_iterator.h`.

4.64.3.23 `template<typename _Iterator, typename _Sequence> reference __gnu_debug::Safe_iterator<_Iterator, _Sequence>::operator* ( ) const [inline]`

Iterator dereference.

**Precondition**

iterator is dereferenceable

Definition at line 260 of file `safe_iterator.h`.

References `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_dereferenceable()`.

4.64.3.24 `template<typename _Iterator, typename _Sequence> _Safe_iterator& __gnu_debug::Safe_iterator<_Iterator, _Sequence>::operator++ ( ) [inline]`

Iterator preincrement.

**Precondition**

iterator is incrementable

Definition at line 289 of file `safe_iterator.h`.

References `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_incrementable()`.

4.64.3.25 `template<typename _Iterator, typename _Sequence> _Safe_iterator __gnu_debug::Safe_iterator<_Iterator, _Sequence>::operator++ ( int ) [inline]`

Iterator postincrement.

**Precondition**

iterator is incrementable

Definition at line 303 of file `safe_iterator.h`.

References `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_incrementable()`.

4.64.3.26 `template<typename _Iterator, typename _Sequence> _Safe_iterator& __gnu_debug::Safe_iterator<_Iterator, _Sequence>::operator-- ( ) [inline]`

Iterator predecrement.

**Precondition**

iterator is decrementable

Definition at line 319 of file `safe_iterator.h`.

4.64.3.27 `template<typename _Iterator, typename _Sequence> _Safe_iterator __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator--( int ) [inline]`

Iterator postdecrement.

#### Precondition

iterator is decrementable

Definition at line 333 of file `safe_iterator.h`.

4.64.3.28 `template<typename _Iterator, typename _Sequence> pointer __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator-> ( ) const [inline]`

Iterator dereference.

#### Precondition

iterator is dereferenceable

**Todo** Make this correct w.r.t. iterators that return proxies

Use `addressof()` instead of `&` operator

Definition at line 275 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::M_dereferenceable()`.

4.64.3.29 `template<typename _Iterator, typename _Sequence> _Safe_iterator& __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator= ( const _Safe_iterator< _Iterator, _Sequence > & __x ) [inline]`

Copy assignment.

Definition at line 217 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::M_attach()`.

4.64.3.30 `template<typename _Iterator, typename _Sequence> _Safe_iterator& __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator= ( _Safe_iterator< _Iterator, _Sequence > && __x ) [inline]`

Move assignment.

#### Postcondition

`__x` is singular and unattached

Definition at line 237 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::M_attach()`.

### 4.64.4 Member Data Documentation

4.64.4.1 `_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::M_next [inherited]`

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 72 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_iterator_base::M_unlink()`.

## 4.64.4.2 \_\_Safe\_iterator\_base\* \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_prior [inherited]

Pointer to the previous iterator in the sequence's list of iterators. Only valid when \_M\_sequence != NULL.

Definition at line 68 of file safe\_base.h.

Referenced by \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_unlink().

## 4.64.4.3 \_\_Safe\_sequence\_base\* \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_sequence [inherited]

The sequence this iterator references; may be NULL to indicate a singular iterator.

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

Referenced by \_\_gnu\_debug::\_Safe\_iterator\_base::\_Safe\_iterator\_base(), and \_\_gnu\_debug::\_Safe\_local\_iterator\_base::\_Safe\_local\_iterator\_base().

## 4.64.4.4 unsigned int \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_version [inherited]

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by \_M\_sequence for the iterator to be non-singular.

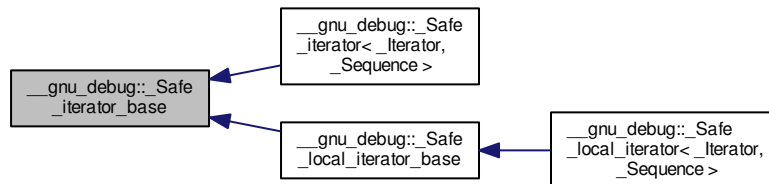
Definition at line 64 of file safe\_base.h.

The documentation for this class was generated from the following files:

- [formatter.h](#)
- [safe\\_iterator.h](#)

## 4.65 \_\_gnu\_debug::\_Safe\_iterator\_base Class Reference

Inheritance diagram for \_\_gnu\_debug::\_Safe\_iterator\_base:



## Public Member Functions

- void [\\_M\\_attach](#) ([\\_Safe\\_sequence\\_base](#) \*\_\_seq, bool \_\_constant)
- void [\\_M\\_attach\\_single](#) ([\\_Safe\\_sequence\\_base](#) \*\_\_seq, bool \_\_constant) throw ()
- bool [\\_M\\_attached\\_to](#) (const [\\_Safe\\_sequence\\_base](#) \*\_\_seq) const
- bool [\\_M\\_can\\_compare](#) (const [\\_Safe\\_iterator\\_base](#) &\_\_x) const throw ()
- void [\\_M\\_detach](#) ()
- void [\\_M\\_detach\\_single](#) () throw ()
- void [\\_M\\_invalidate](#) ()



- void `_M_reset()` throw ()
- bool `_M_singular()` const throw ()
- void `_M_unlink()` throw ()

#### Public Attributes

- `_Safe_iterator_base * _M_next`
- `_Safe_iterator_base * _M_prior`
- `_Safe_sequence_base * _M_sequence`
- unsigned int `_M_version`

#### Protected Member Functions

- `_Safe_iterator_base()`
- `_Safe_iterator_base(const _Safe_sequence_base * __seq, bool __constant)`
- `_Safe_iterator_base(const _Safe_iterator_base & __x, bool __constant)`
- `_Safe_iterator_base(const _Safe_iterator_base &)`
- `__gnu_cxx::__mutex & _M_get_mutex()` throw ()
- `_Safe_iterator_base & operator= (const _Safe_iterator_base &)`

#### 4.65.1 Detailed Description

Basic functionality for a *safe* iterator.

The `_Safe_iterator_base` base class implements the functionality of a safe iterator that is not specific to a particular iterator type. It contains a pointer back to the sequence it references along with iterator version information and pointers to form a doubly-linked list of iterators referenced by the container.

This class must not perform any operations that can throw an exception, or the exception guarantees of derived iterators will be broken.

Definition at line 50 of file `safe_base.h`.

#### 4.65.2 Constructor & Destructor Documentation

##### 4.65.2.1 `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base ( )` `[inline]`, `[protected]`

Initializes the iterator and makes it singular.

Definition at line 76 of file `safe_base.h`.

##### 4.65.2.2 `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base ( const _Safe_sequence_base * __seq, bool __constant )` `[inline]`, `[protected]`

Initialize the iterator to reference the sequence pointed to by `__seq`. `__constant` is true when we are initializing a constant iterator, and false if it is a mutable iterator. Note that `__seq` may be NULL, in which case the iterator will be singular. Otherwise, the iterator will reference `__seq` and be nonsingular.

Definition at line 87 of file `safe_base.h`.

References `_M_attach()`.

4.65.2.3 `__gnu_debug::Safe_iterator_base::Safe_iterator_base ( const _Safe_iterator_base & __x, bool __constant )`  
`[inline], [protected]`

Initializes the iterator to reference the same sequence that `__x` does. `__constant` is true if this is a constant iterator, and false if it is mutable.

Definition at line 94 of file `safe_base.h`.

References `_M_attach()`, and `_M_sequence`.

### 4.65.3 Member Function Documentation

4.65.3.1 `void __gnu_debug::Safe_iterator_base::M_attach ( _Safe_sequence_base * __seq, bool __constant )`

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

Referenced by `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >::M_attach()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_attach()`, and `_Safe_iterator_base()`.

4.65.3.2 `void __gnu_debug::Safe_iterator_base::M_attach_single ( _Safe_sequence_base * __seq, bool __constant ) throw ()`

Likewise, but not thread-safe.

Referenced by `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >::M_attach_single()`, and `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_attach_single()`.

4.65.3.3 `bool __gnu_debug::Safe_iterator_base::M_attached_to ( const _Safe_sequence_base * __seq ) const`  
`[inline]`

Determines if we are attached to the given sequence.

Definition at line 129 of file `safe_base.h`.

4.65.3.4 `bool __gnu_debug::Safe_iterator_base::M_can_compare ( const _Safe_iterator_base & __x ) const throw ()`

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

4.65.3.5 `void __gnu_debug::Safe_iterator_base::M_detach ( )`

Detach the iterator for whatever sequence it is attached to, if any.

4.65.3.6 `void __gnu_debug::Safe_iterator_base::M_detach_single ( ) throw ()`

Likewise, but not thread-safe.

4.65.3.7 `__gnu_cxx::mutex& __gnu_debug::Safe_iterator_base::M_get_mutex ( ) throw ()` `[protected]`

For use in `_Safe_iterator`.

4.65.3.8 `void __gnu_debug::Safe_iterator_base::M_invalidate ( )` `[inline]`

Invalidate the iterator, making it singular.

Definition at line 142 of file `safe_base.h`.

4.65.3.9 `void __gnu_debug::Safe_iterator_base::_M_reset ( ) throw`

Reset all member variables

4.65.3.10 `bool __gnu_debug::Safe_iterator_base::_M_singular ( ) const throw`

Is this iterator singular?

Referenced by `__gnu_debug::check_singular()`, `__gnu_debug::check_singular_aux()`, `__gnu_debug::Safe_iterator_base::_M_dereferenceable()`, `__gnu_debug::Safe_iterator_base::_M_dereferenceable()`, `__gnu_debug::Safe_iterator_base::_M_incrementable()`, `__gnu_debug::Safe_iterator_base::_M_incrementable()`, `__gnu_debug::Safe_iterator_base::_M_incrementable()`, `__gnu_debug::Safe_iterator_base::_M_incrementable()`, `__gnu_debug::Safe_iterator_base::_M_incrementable()`, and `__gnu_debug::Safe_iterator_base::_M_incrementable()`.

4.65.3.11 `void __gnu_debug::Safe_iterator_base::_M_unlink ( ) throw` `[inline]`

Unlink itself

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

References `_M_next`, and `_M_prior`.

#### 4.65.4 Member Data Documentation

4.65.4.1 `Safe_iterator_base* __gnu_debug::Safe_iterator_base::_M_next`

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 72 of file `safe_base.h`.

Referenced by `_M_unlink()`.

4.65.4.2 `Safe_iterator_base* __gnu_debug::Safe_iterator_base::_M_prior`

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 68 of file `safe_base.h`.

Referenced by `_M_unlink()`.

4.65.4.3 `Safe_sequence_base* __gnu_debug::Safe_iterator_base::_M_sequence`

The sequence this iterator references; may be `NULL` to indicate a singular iterator.

Definition at line 55 of file `safe_base.h`.

Referenced by `_Safe_iterator_base()`, and `__gnu_debug::Safe_iterator_base::_Safe_iterator_base()`.

4.65.4.4 `unsigned int __gnu_debug::Safe_iterator_base::_M_version`

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

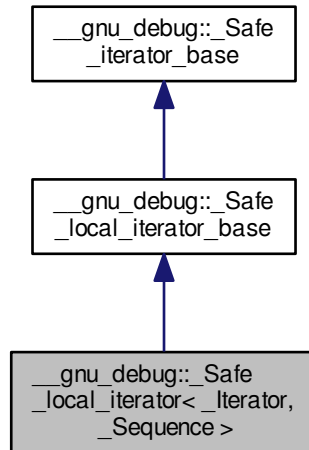
Definition at line 64 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [safe\\_base.h](#)

4.66 `__gnu_debug::__Safe_local_iterator<_Iterator, _Sequence>` Class Template Reference

Inheritance diagram for `__gnu_debug::__Safe_local_iterator<_Iterator, _Sequence>`:



## Public Types

- `typedef _Traits::difference_type` **difference\_type**
- `typedef _Traits::iterator_category` **iterator\_category**
- `typedef _Iterator` **iterator\_type**
- `typedef _Traits::pointer` **pointer**
- `typedef _Traits::reference` **reference**
- `typedef _Traits::value_type` **value\_type**

## Public Member Functions

- `_Safe_local_iterator` ()
- `_Safe_local_iterator` (const `_Iterator` &\_\_i, `size_type` \_\_bucket, const `_Sequence` \*\_\_seq)
- `_Safe_local_iterator` (const `_Safe_local_iterator` &\_\_x)
- `template<typename _MutableIterator>` `_Safe_local_iterator` (const `_Safe_local_iterator`< `_MutableIterator`, `typename` `__gnu_cxx::__enable_if`< `std::__are_same`< `_MutableIterator`, `typename` `_Sequence::local_iterator::iterator`< `type`>::`_value`, `_Sequence`>::`_type`> &\_\_x)
- `void` `_M_attach` (`_Safe_sequence_base` \*\_\_seq, bool \_\_constant)
- `void` `_M_attach` (`_Safe_sequence_base` \*\_\_seq)
- `void` `_M_attach_single` (`_Safe_sequence_base` \*\_\_seq, bool \_\_constant) throw ()
- `void` `_M_attach_single` (`_Safe_sequence_base` \*\_\_seq)
- `bool` `_M_attached_to` (const `_Safe_sequence_base` \*\_\_seq) const
- `bool` `_M_can_compare` (const `_Safe_iterator_base` &\_\_x) const throw ()
- `bool` `_M_dereferenceable` () const
- `void` `_M_detach` ()

- void `_M_detach_single` () throw ()
- const `_Sequence` \* `_M_get_sequence` () const
- template<typename `_Other` > bool `_M_in_same_bucket` (const `_Safe_local_iterator`< `_Other`, `_Sequence` > &\_\_other) const
- bool `_M_incrementable` () const
- void `_M_invalidate` ()
- bool `_M_is_begin` () const
- bool `_M_is_end` () const
- void `_M_reset` () throw ()
- bool `_M_singular` () const throw ()
- void `_M_unlink` () throw ()
- template<typename `_Other` > bool `_M_valid_range` (const `_Safe_local_iterator`< `_Other`, `_Sequence` > &\_\_rhs) const
- `_Iterator` `base` () const
- size\_type `bucket` () const
- `operator _Iterator` () const
- reference `operator*` () const
- `_Safe_local_iterator` & `operator++` ()
- `_Safe_local_iterator` `operator++` (int)
- pointer `operator->` () const
- `_Safe_local_iterator` & `operator=` (const `_Safe_local_iterator` &\_\_x)

#### Public Attributes

- `_Safe_iterator_base` \* `_M_next`
- `_Safe_iterator_base` \* `_M_prior`
- `_Safe_sequence_base` \* `_M_sequence`
- unsigned int `_M_version`

#### Protected Member Functions

- `_Safe_unordered_container_base` \* `_M_get_container` () const noexcept
- `__gnu_cxx::__mutex` & `_M_get_mutex` () throw ()

#### 4.66.1 Detailed Description

```
template<typename _Iterator, typename _Sequence>class __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >
```

Safe iterator wrapper.

The class template `_Safe_local_iterator` is a wrapper around an iterator that tracks the iterator's movement among sequences and checks that operations performed on the "safe" iterator are legal. In addition to the basic iterator operations (which are validated, and then passed to the underlying iterator), `_Safe_local_iterator` has member functions for iterator invalidation, attaching/detaching the iterator from sequences, and querying the iterator's state.

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

## 4.66.2 Constructor &amp; Destructor Documentation

4.66.2.1 `template<typename _Iterator, typename _Sequence> __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::__Safe_local_iterator ( )` `[inline]`

## Postcondition

the iterator is singular and unattached

Definition at line 82 of file `safe_local_iterator.h`.

4.66.2.2 `template<typename _Iterator, typename _Sequence> __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::__Safe_local_iterator ( const _Iterator & __i, size_type __bucket, const _Sequence * __seq )` `[inline]`

Safe iterator construction from an unsafe iterator and its sequence.

## Precondition

`seq` is not NULL

## Postcondition

this is not singular

Definition at line 91 of file `safe_local_iterator.h`.

References `__gnu_debug::Safe_iterator_base::M_singular()`.

4.66.2.3 `template<typename _Iterator, typename _Sequence> __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::__Safe_local_iterator ( const _Safe_local_iterator<_Iterator, _Sequence> & __x )` `[inline]`

Copy construction.

Definition at line 104 of file `safe_local_iterator.h`.

4.66.2.4 `template<typename _Iterator, typename _Sequence> template<typename _MutableIterator > __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::__Safe_local_iterator ( const _Safe_local_iterator<_MutableIterator, typename __gnu_cxx::__enable_if<std::__are_same<_MutableIterator, typename _Sequence::local_iterator::iterator_type>::__value, _Sequence>::__type> & __x )` `[inline]`

Converting constructor from a mutable iterator to a constant iterator.

Definition at line 122 of file `safe_local_iterator.h`.

## 4.66.3 Member Function Documentation

4.66.3.1 `void __gnu_debug::Safe_local_iterator_base::M_attach ( _Safe_sequence_base * __seq, bool __constant )` `[inherited]`

Attaches this iterator to the given container, detaching it from whatever container it was attached to originally. If the new container is the NULL pointer, the iterator is left unattached.

Referenced by `__gnu_debug::Safe_local_iterator_base::Safe_local_iterator_base()`.

4.66.3.2 `template<typename _Iterator, typename _Sequence> void __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::__M_attach ( _Safe_sequence_base * __seq )` `[inline]`

Attach iterator to the given sequence.

Definition at line 238 of file `safe_local_iterator.h`.

References `__gnu_debug::Safe_iterator_base::_M_attach()`.

Referenced by `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::operator=()`.

4.66.3.3 `void __gnu_debug::Safe_local_iterator_base::_M_attach_single ( _Safe_sequence_base * __seq, bool __constant ) throw ) [inherited]`

Likewise, but not thread-safe.

4.66.3.4 `template<typename _Iterator, typename _Sequence> void __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::_M_attach_single ( _Safe_sequence_base * __seq ) [inline]`

Likewise, but not thread-safe.

Definition at line 243 of file `safe_local_iterator.h`.

References `__gnu_debug::Safe_iterator_base::_M_attach_single()`.

4.66.3.5 `bool __gnu_debug::Safe_iterator_base::_M_attached_to ( const _Safe_sequence_base * __seq ) const [inline], [inherited]`

Determines if we are attached to the given sequence.

Definition at line 129 of file `safe_base.h`.

4.66.3.6 `bool __gnu_debug::Safe_iterator_base::_M_can_compare ( const _Safe_iterator_base & __x ) const throw ) [inherited]`

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

4.66.3.7 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::_M_dereferenceable ( ) const [inline]`

Is the iterator dereferenceable?

Definition at line 248 of file `safe_local_iterator.h`.

References `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::_M_is_end()`, and `__gnu_debug::Safe_iterator_base::_M_singular()`.

Referenced by `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::operator*()`, and `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::operator->()`.

4.66.3.8 `void __gnu_debug::Safe_local_iterator_base::_M_detach ( ) [inherited]`

Detach the iterator for whatever container it is attached to, if any.

4.66.3.9 `void __gnu_debug::Safe_local_iterator_base::_M_detach_single ( ) throw ) [inherited]`

Likewise, but not thread-safe.

4.66.3.10 `__gnu_cxx::mutex& __gnu_debug::Safe_iterator_base::_M_get_mutex ( ) throw ) [protected], [inherited]`

For use in `_Safe_iterator`.

4.66.3.11 `template<typename _Iterator, typename _Sequence> template<typename _Other> bool __gnu_debug::Safe_↵  
local_iterator<_Iterator, _Sequence>::__M_in_same_bucket ( const _Safe_local_iterator<_Other, _Sequence>  
&__other ) const [inline]`

Is this iterator part of the same bucket as the other one?

Definition at line 277 of file `safe_local_iterator.h`.

4.66.3.12 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_local_iterator<_Iterator,  
_Sequence>::__M_incrementable ( ) const [inline]`

Is the iterator incrementable?

Definition at line 253 of file `safe_local_iterator.h`.

References `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::__M_is_end()`, and `__gnu_debug::Safe_↵  
iterator_base::__M_singular()`.

Referenced by `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::operator++()`.

4.66.3.13 `void __gnu_debug::Safe_iterator_base::__M_invalidate ( ) [inline],[inherited]`

Invalidate the iterator, making it singular.

Definition at line 142 of file `safe_base.h`.

4.66.3.14 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_local_iterator<_Iterator,  
_Sequence>::__M_is_begin ( ) const [inline]`

Is this iterator equal to the sequence's `begin()` iterator?

Definition at line 268 of file `safe_local_iterator.h`.

References `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::base()`.

4.66.3.15 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_local_iterator<_Iterator,  
_Sequence>::__M_is_end ( ) const [inline]`

Is this iterator equal to the sequence's `end()` iterator?

Definition at line 272 of file `safe_local_iterator.h`.

References `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::base()`.

Referenced by `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::__M_dereferenceable()`, and `__gnu_↵  
debug::Safe_local_iterator<_Iterator, _Sequence>::__M_incrementable()`.

4.66.3.16 `void __gnu_debug::Safe_iterator_base::__M_reset ( ) throw ) [inherited]`

Reset all member variables

4.66.3.17 `bool __gnu_debug::Safe_iterator_base::__M_singular ( ) const throw ) [inherited]`

Is this iterator singular?

Referenced by `__gnu_debug::check_singular()`, `__gnu_debug::check_singular_aux()`, `__gnu_debug::Safe_↵  
local_iterator<_Iterator, _Sequence>::__M_dereferenceable()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>  
>::__M_dereferenceable()`, `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::__M_incrementable()`, `__↵  
gnu_debug::Safe_iterator<_Iterator, _Sequence>::__M_incrementable()`, `__gnu_debug::Safe_iterator<_Iterator,  
_Sequence>::__Safe_iterator()`, and `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::__Safe_local_↵  
iterator()`.



4.66.3.18 `void __gnu_debug::_Safe_iterator_base::_M_unlink ( ) throw` `[inline],[inherited]`

Unlink itself

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

References `__gnu_debug::_Safe_iterator_base::_M_next`, and `__gnu_debug::_Safe_iterator_base::_M_prior`.

4.66.3.19 `template<typename _Iterator, typename _Sequence> _Iterator __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::base ( ) const` `[inline]`

Return the underlying iterator.

Definition at line 222 of file `safe_local_iterator.h`.

Referenced by `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_is_begin()`, and `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_is_end()`.

4.66.3.20 `template<typename _Iterator, typename _Sequence> size_type __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::bucket ( ) const` `[inline]`

Return the bucket.

Definition at line 228 of file `safe_local_iterator.h`.

4.66.3.21 `template<typename _Iterator, typename _Sequence> __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator _Iterator ( ) const` `[inline]`

Conversion to underlying non-debug iterator to allow better interaction with non-debug containers.

Definition at line 234 of file `safe_local_iterator.h`.

4.66.3.22 `template<typename _Iterator, typename _Sequence> reference __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator* ( ) const` `[inline]`

Iterator dereference.

Precondition

iterator is dereferenceable

Definition at line 164 of file `safe_local_iterator.h`.

References `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_dereferenceable()`.

4.66.3.23 `template<typename _Iterator, typename _Sequence> _Safe_local_iterator& __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator++ ( )` `[inline]`

Iterator preincrement.

Precondition

iterator is incrementable

Definition at line 193 of file `safe_local_iterator.h`.

References `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_incrementable()`.

4.66.3.24 `template<typename _Iterator, typename _Sequence> _Safe_local_iterator __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator++ ( int )` `[inline]`

Iterator postincrement.

**Precondition**

iterator is incrementable

Definition at line 207 of file `safe_local_iterator.h`.

References `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::_M_incrementable()`.

4.66.3.25 `template<typename _Iterator, typename _Sequence> pointer __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::operator-> ( ) const [inline]`

Iterator dereference.

**Precondition**

iterator is dereferenceable

**Todo** Make this correct w.r.t. iterators that return proxies

Use `addressof()` instead of `&` operator

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

References `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::_M_dereferenceable()`.

4.66.3.26 `template<typename _Iterator, typename _Sequence> _Safe_local_iterator& __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::operator= ( const _Safe_local_iterator<_Iterator, _Sequence> &_x ) [inline]`

Copy assignment.

Definition at line 144 of file `safe_local_iterator.h`.

References `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::_M_attach()`.

**4.66.4 Member Data Documentation**

4.66.4.1 `_Safe_iterator_base* __gnu_debug::Safe_iterator_base::_M_next [inherited]`

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 72 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_iterator_base::_M_unlink()`.

4.66.4.2 `_Safe_iterator_base* __gnu_debug::Safe_iterator_base::_M_prior [inherited]`

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 68 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_iterator_base::_M_unlink()`.

4.66.4.3 `_Safe_sequence_base* __gnu_debug::Safe_iterator_base::_M_sequence [inherited]`

The sequence this iterator references; may be `NULL` to indicate a singular iterator.

Definition at line 55 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_iterator_base::_Safe_iterator_base()`, and `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::_Safe_local_iterator_base()`.

#### 4.66.4.4 unsigned int \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_version [inherited]

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by \_M\_sequence for the iterator to be non-singular.

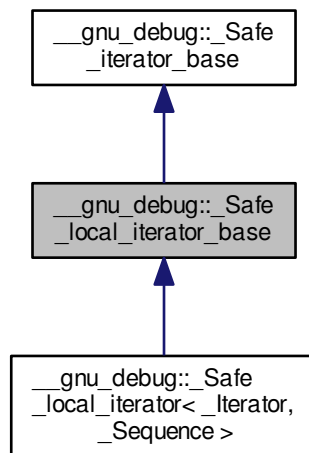
Definition at line 64 of file safe\_base.h.

The documentation for this class was generated from the following files:

- [formatter.h](#)
- [safe\\_local\\_iterator.h](#)

#### 4.67 \_\_gnu\_debug::\_Safe\_local\_iterator\_base Class Reference

Inheritance diagram for \_\_gnu\_debug::\_Safe\_local\_iterator\_base:



#### Public Member Functions

- void [\\_M\\_attach](#) ([\\_Safe\\_sequence\\_base](#) \*\_\_seq, bool \_\_constant)
- void [\\_M\\_attach\\_single](#) ([\\_Safe\\_sequence\\_base](#) \*\_\_seq, bool \_\_constant) throw ()
- bool [\\_M\\_attached\\_to](#) (const [\\_Safe\\_sequence\\_base](#) \*\_\_seq) const
- bool [\\_M\\_can\\_compare](#) (const [\\_Safe\\_iterator\\_base](#) &\_\_x) const throw ()
- void [\\_M\\_detach](#) ()
- void [\\_M\\_detach\\_single](#) () throw ()
- void [\\_M\\_invalidate](#) ()
- void [\\_M\\_reset](#) () throw ()
- bool [\\_M\\_singular](#) () const throw ()
- void [\\_M\\_unlink](#) () throw ()

## Public Attributes

- `_Safe_iterator_base * _M_next`
- `_Safe_iterator_base * _M_prior`
- `_Safe_sequence_base * _M_sequence`
- unsigned int `_M_version`

## Protected Member Functions

- `_Safe_local_iterator_base ()`
- `_Safe_local_iterator_base (const _Safe_sequence_base * __seq, bool __constant)`
- `_Safe_local_iterator_base (const _Safe_local_iterator_base & __x, bool __constant)`
- `_Safe_local_iterator_base (const _Safe_local_iterator_base &)`
- `_Safe_unordered_container_base * _M_get_container ()` const noexcept
- `__gnu_cxx::mutex & _M_get_mutex ()` throw ()
- `_Safe_local_iterator_base & operator= (const _Safe_local_iterator_base &)`

## 4.67.1 Detailed Description

Basic functionality for a *safe* iterator.

The `_Safe_local_iterator_base` base class implements the functionality of a safe local iterator that is not specific to a particular iterator type. It contains a pointer back to the container it references along with iterator version information and pointers to form a doubly-linked list of local iterators referenced by the container.

This class must not perform any operations that can throw an exception, or the exception guarantees of derived iterators will be broken.

Definition at line 50 of file `safe_unordered_base.h`.

## 4.67.2 Constructor &amp; Destructor Documentation

4.67.2.1 `__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base ( )` `[inline]`, `[protected]`

Initializes the iterator and makes it singular.

Definition at line 54 of file `safe_unordered_base.h`.

4.67.2.2 `__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base ( const _Safe_sequence_base * __seq, bool __constant )` `[inline]`, `[protected]`

Initialize the iterator to reference the container pointed to by `__seq`. `__constant` is true when we are initializing a constant local iterator, and false if it is a mutable local iterator. Note that `__seq` may be NULL, in which case the iterator will be singular. Otherwise, the iterator will reference `__seq` and be nonsingular.

Definition at line 64 of file `safe_unordered_base.h`.

References `_M_attach()`.

4.67.2.3 `__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base ( const _Safe_local_iterator_base & __x, bool __constant )` `[inline]`, `[protected]`

Initializes the iterator to reference the same container that `__x` does. `__constant` is true if this is a constant iterator, and false if it is mutable.

Definition at line 70 of file `safe_unordered_base.h`.

References `_M_attach()`, and `__gnu_debug::_Safe_iterator_base::_M_sequence`.

#### 4.67.3 Member Function Documentation

4.67.3.1 `void __gnu_debug::_Safe_local_iterator_base::_M_attach ( _Safe_sequence_base * __seq, bool __constant )`

Attaches this iterator to the given container, detaching it from whatever container it was attached to originally. If the new container is the NULL pointer, the iterator is left unattached.

Referenced by `_Safe_local_iterator_base()`.

4.67.3.2 `void __gnu_debug::_Safe_local_iterator_base::_M_attach_single ( _Safe_sequence_base * __seq, bool __constant ) throw )`

Likewise, but not thread-safe.

4.67.3.3 `bool __gnu_debug::_Safe_iterator_base::_M_attached_to ( const _Safe_sequence_base * __seq ) const [inline], [inherited]`

Determines if we are attached to the given sequence.

Definition at line 129 of file `safe_base.h`.

4.67.3.4 `bool __gnu_debug::_Safe_iterator_base::_M_can_compare ( const _Safe_iterator_base & __x ) const throw ) [inherited]`

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

4.67.3.5 `void __gnu_debug::_Safe_local_iterator_base::_M_detach ( )`

Detach the iterator for whatever container it is attached to, if any.

4.67.3.6 `void __gnu_debug::_Safe_local_iterator_base::_M_detach_single ( ) throw )`

Likewise, but not thread-safe.

4.67.3.7 `__gnu_cxx::mutex& __gnu_debug::_Safe_iterator_base::_M_get_mutex ( ) throw ) [protected], [inherited]`

For use in `_Safe_iterator`.

4.67.3.8 `void __gnu_debug::_Safe_iterator_base::_M_invalidate ( ) [inline], [inherited]`

Invalidate the iterator, making it singular.

Definition at line 142 of file `safe_base.h`.

4.67.3.9 `void __gnu_debug::_Safe_iterator_base::_M_reset ( ) throw ) [inherited]`

Reset all member variables

4.67.3.10 `bool __gnu_debug::_Safe_iterator_base::_M_singular ( ) const throw ) [inherited]`

Is this iterator singular?

Referenced by `__gnu_debug::_check_singular()`, `__gnu_debug::_check_singular_aux()`, `__gnu_debug::_Safe_↵ local_iterator< _Iterator, _Sequence >::_M_dereferenceable()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence`

>::\_M\_dereferenceable(), \_\_gnu\_debug::\_Safe\_local\_iterator< \_Iterator, \_Sequence >::\_M\_incrementable(), \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence >::\_M\_incrementable(), \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence >::\_Safe\_iterator(), and \_\_gnu\_debug::\_Safe\_local\_iterator< \_Iterator, \_Sequence >::\_Safe\_local\_iterator().

4.67.3.11 void \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_unlink ( ) throw [inline],[inherited]

Unlink itself

Definition at line 151 of file safe\_base.h.

References \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_next, and \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_prior.

#### 4.67.4 Member Data Documentation

4.67.4.1 \_Safe\_iterator\_base\* \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_next [inherited]

Pointer to the next iterator in the sequence's list of iterators. Only valid when \_M\_sequence != NULL.

Definition at line 72 of file safe\_base.h.

Referenced by \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_unlink().

4.67.4.2 \_Safe\_iterator\_base\* \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_prior [inherited]

Pointer to the previous iterator in the sequence's list of iterators. Only valid when \_M\_sequence != NULL.

Definition at line 68 of file safe\_base.h.

Referenced by \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_unlink().

4.67.4.3 \_Safe\_sequence\_base\* \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_sequence [inherited]

The sequence this iterator references; may be NULL to indicate a singular iterator.

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

Referenced by \_\_gnu\_debug::\_Safe\_iterator\_base::\_Safe\_iterator\_base(), and \_Safe\_local\_iterator\_base().

4.67.4.4 unsigned int \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_version [inherited]

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by \_M\_sequence for the iterator to be non-singular.

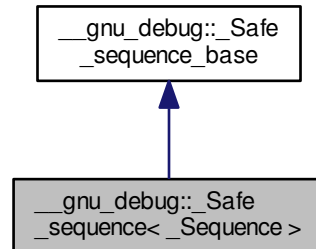
Definition at line 64 of file safe\_base.h.

The documentation for this class was generated from the following file:

- [safe\\_unordered\\_base.h](#)

#### 4.68 `__gnu_debug::Safe_sequence<_Sequence>` Class Template Reference

Inheritance diagram for `__gnu_debug::Safe_sequence<_Sequence>`:



##### Public Member Functions

- `void _M_attach (_Safe_iterator_base * __it, bool __constant)`
- `void _M_attach_single (_Safe_iterator_base * __it, bool __constant) throw ()`
- `void _M_detach (_Safe_iterator_base * __it)`
- `void _M_detach_single (_Safe_iterator_base * __it) throw ()`
- `void _M_invalidate_all () const`
- `template<typename _Predicate> void _M_invalidate_if (_Predicate __pred)`
- `template<typename _Predicate> void _M_transfer_from_if (_Safe_sequence & __from, _Predicate __pred)`

##### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

##### Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_sequence_base & __x)`

##### 4.68.1 Detailed Description

```
template<typename _Sequence> class __gnu_debug::Safe_sequence<_Sequence>
```

Base class for constructing a *safe* sequence type that tracks iterators that reference it.

The class template `_Safe_sequence` simplifies the construction of *safe* sequences that track the iterators that reference the sequence, so that the iterators are notified of changes in the sequence that may affect their operation, e.g., if the container invalidates its iterators or is destructed. This class template may only be used by deriving from it and passing the name of the derived class as its template parameter via the curiously recurring template pattern. The derived class must have `iterator` and `const_iterator` types that are instantiations of class template `_Safe_iterator` for this sequence. Iterators will then be tracked automatically.

Definition at line 52 of file `formatter.h`.

#### 4.68.2 Member Function Documentation

**4.68.2.1** `void __gnu_debug::_Safe_sequence_base::M_attach ( _Safe_iterator_base * __it, bool __constant )`  
[*inherited*]

Attach an iterator to this sequence.

**4.68.2.2** `void __gnu_debug::_Safe_sequence_base::M_attach_single ( _Safe_iterator_base * __it, bool __constant ) throw )`  
[*inherited*]

Likewise but not thread safe.

**4.68.2.3** `void __gnu_debug::_Safe_sequence_base::M_detach ( _Safe_iterator_base * __it )` [*inherited*]

Detach an iterator from this sequence

**4.68.2.4** `void __gnu_debug::_Safe_sequence_base::M_detach_all ( )` [*protected*],[*inherited*]

Detach all iterators, leaving them singular.

Referenced by `__gnu_debug::_Safe_sequence_base::~~_Safe_sequence_base()`.

**4.68.2.5** `void __gnu_debug::_Safe_sequence_base::M_detach_single ( _Safe_iterator_base * __it ) throw )`  
[*inherited*]

Likewise but not thread safe.

**4.68.2.6** `void __gnu_debug::_Safe_sequence_base::M_detach_singular ( )` [*protected*],[*inherited*]

Detach all singular iterators.

#### Postcondition

for all iterators *i* attached to this sequence, `i->_M_version == _M_version`.

**4.68.2.7** `__gnu_cxx::mutex& __gnu_debug::_Safe_sequence_base::M_get_mutex ( ) throw )` [*protected*],[*inherited*]

For use in `_Safe_sequence`.

**4.68.2.8** `void __gnu_debug::_Safe_sequence_base::M_invalidate_all ( ) const` [*inline*],[*inherited*]

Invalidates all iterators.

Definition at line 233 of file `safe_base.h`.



4.68.2.9 `template<typename _Sequence> template<typename _Predicate > void __gnu_debug::_Safe_sequence<_Sequence >::_M_invalidate_if ( _Predicate __pred )`

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.68.2.10 `void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected],[inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.68.2.11 `void __gnu_debug::_Safe_sequence_base::_M_swap ( _Safe_sequence_base & __x ) [protected],[inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.68.2.12 `template<typename _Sequence> template<typename _Predicate > void __gnu_debug::_Safe_sequence<_Sequence >::_M_transfer_from_if ( _Safe_sequence<_Sequence > & __from, _Predicate __pred )`

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

### 4.68.3 Member Data Documentation

4.68.3.1 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]`

The list of constant iterators that reference this container.

Definition at line 184 of file `safe_base.h`.

4.68.3.2 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]`

The list of mutable iterators that reference this container.

Definition at line 181 of file `safe_base.h`.

4.68.3.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable],[inherited]`

The container version number. This number may never be 0.

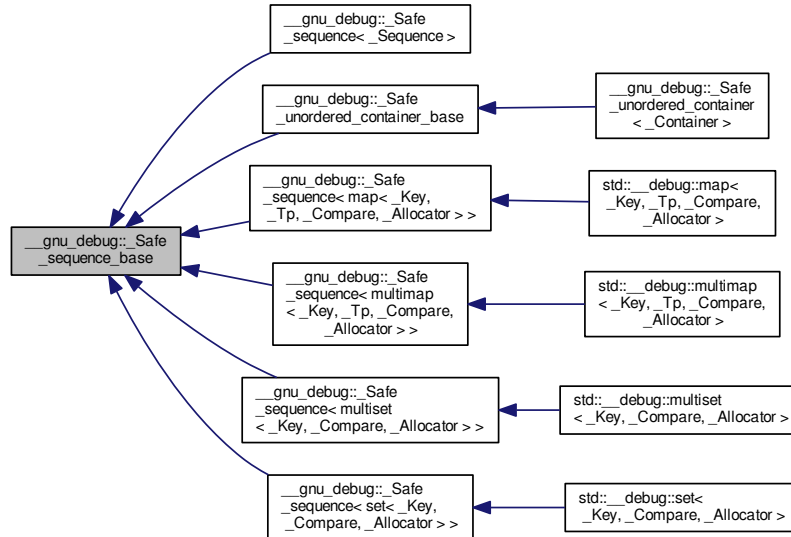
Definition at line 187 of file `safe_base.h`.

The documentation for this class was generated from the following files:

- [formatter.h](#)
- [safe\\_sequence.h](#)

## 4.69 \_\_gnu\_debug::\_\_Safe\_sequence\_base Class Reference

Inheritance diagram for \_\_gnu\_debug::\_\_Safe\_sequence\_base:



## Public Member Functions

- void [\\_M\\_attach](#) ([\\_Safe\\_iterator\\_base](#) \* \_\_it, bool \_\_constant)
- void [\\_M\\_attach\\_single](#) ([\\_Safe\\_iterator\\_base](#) \* \_\_it, bool \_\_constant) throw ()
- void [\\_M\\_detach](#) ([\\_Safe\\_iterator\\_base](#) \* \_\_it)
- void [\\_M\\_detach\\_single](#) ([\\_Safe\\_iterator\\_base](#) \* \_\_it) throw ()
- void [\\_M\\_invalidate\\_all](#) () const

## Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_const\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_iterators](#)
- unsigned int [\\_M\\_version](#)

## Protected Member Functions

- [~\\_Safe\\_sequence\\_base](#) ()
- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_revalidate\\_singular](#) ()
- void [\\_M\\_swap](#) ([\\_Safe\\_sequence\\_base](#) & \_\_x)

#### 4.69.1 Detailed Description

Base class that supports tracking of iterators that reference a sequence.

The `_Safe_sequence_base` class provides basic support for tracking iterators into a sequence. Sequences that track iterators must derived from `_Safe_sequence_base` publicly, so that safe iterators (which inherit `_Safe_iterator_base`) can attach to them. This class contains two linked lists of iterators, one for constant iterators and one for mutable iterators, and a version number that allows very fast invalidation of all iterators that reference the container.

This class must ensure that no operation on it may throw an exception, otherwise *safe* sequences may fail to provide the exception-safety guarantees required by the C++ standard.

Definition at line 177 of file `safe_base.h`.

#### 4.69.2 Constructor & Destructor Documentation

##### 4.69.2.1 `__gnu_debug::_Safe_sequence_base::~~_Safe_sequence_base ( )` `[inline]`, `[protected]`

Notify all iterators that reference this sequence that the sequence is being destroyed.

Definition at line 197 of file `safe_base.h`.

References `_M_detach_all()`.

#### 4.69.3 Member Function Documentation

##### 4.69.3.1 `void __gnu_debug::_Safe_sequence_base::_M_attach ( _Safe_iterator_base * __it, bool __constant )`

Attach an iterator to this sequence.

##### 4.69.3.2 `void __gnu_debug::_Safe_sequence_base::_M_attach_single ( _Safe_iterator_base * __it, bool __constant ) throw`

Likewise but not thread safe.

##### 4.69.3.3 `void __gnu_debug::_Safe_sequence_base::_M_detach ( _Safe_iterator_base * __it )`

Detach an iterator from this sequence

##### 4.69.3.4 `void __gnu_debug::_Safe_sequence_base::_M_detach_all ( )` `[protected]`

Detach all iterators, leaving them singular.

Referenced by `~_Safe_sequence_base()`.

##### 4.69.3.5 `void __gnu_debug::_Safe_sequence_base::_M_detach_single ( _Safe_iterator_base * __it ) throw`

Likewise but not thread safe.

##### 4.69.3.6 `void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( )` `[protected]`

Detach all singular iterators.

#### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

4.69.3.7 `__gnu_cxx::mutex& __gnu_debug::Safe_sequence_base::M_get_mutex ( ) throw` [protected]

For use in `_Safe_sequence`.

4.69.3.8 `void __gnu_debug::Safe_sequence_base::M_invalidate_all ( ) const` [inline]

Invalidates all iterators.

Definition at line 233 of file `safe_base.h`.

4.69.3.9 `void __gnu_debug::Safe_sequence_base::M_revalidate_singular ( )` [protected]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.69.3.10 `void __gnu_debug::Safe_sequence_base::M_swap ( _Safe_sequence_base & __x )` [protected]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

#### 4.69.4 Member Data Documentation

4.69.4.1 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_const_iterators`

The list of constant iterators that reference this container.

Definition at line 184 of file `safe_base.h`.

4.69.4.2 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_iterators`

The list of mutable iterators that reference this container.

Definition at line 181 of file `safe_base.h`.

4.69.4.3 `unsigned int __gnu_debug::Safe_sequence_base::M_version` [mutable]

The container version number. This number may never be 0.

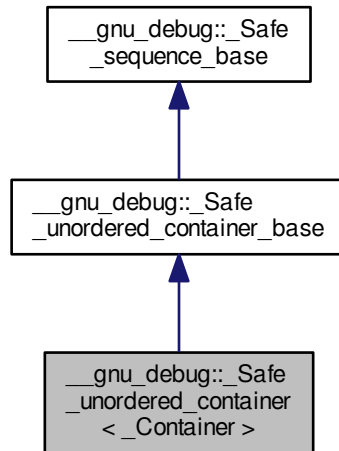
Definition at line 187 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [safe\\_base.h](#)

#### 4.70 `__gnu_debug::Safe_unordered_container<_Container>` Class Template Reference

Inheritance diagram for `__gnu_debug::Safe_unordered_container<_Container>`:



##### Public Member Functions

- `void _M_attach (_Safe_iterator_base * __it, bool __constant)`
- `void _M_attach_local (_Safe_iterator_base * __it, bool __constant)`
- `void _M_attach_local_single (_Safe_iterator_base * __it, bool __constant) throw ()`
- `void _M_attach_single (_Safe_iterator_base * __it, bool __constant) throw ()`
- `void _M_detach (_Safe_iterator_base * __it)`
- `void _M_detach_local (_Safe_iterator_base * __it)`
- `void _M_detach_local_single (_Safe_iterator_base * __it) throw ()`
- `void _M_detach_single (_Safe_iterator_base * __it) throw ()`
- `void _M_invalidate_all () const`
- `template<typename _Predicate> void _M_invalidate_if (_Predicate __pred)`
- `template<typename _Predicate> void _M_invalidate_local_if (_Predicate __pred)`

##### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_const_local_iterators`
- `_Safe_iterator_base * _M_iterators`
- `_Safe_iterator_base * _M_local_iterators`
- `unsigned int _M_version`

## Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_unordered_container_base & __x)`
- `void _M_swap (_Safe_sequence_base & __x)`

## 4.70.1 Detailed Description

```
template<typename _Container>class __gnu_debug::_Safe_unordered_container<_Container>
```

Base class for constructing a *safe* unordered container type that tracks iterators that reference it.

The class template `_Safe_unordered_container` simplifies the construction of *safe* unordered containers that track the iterators that reference the container, so that the iterators are notified of changes in the container that may affect their operation, e.g., if the container invalidates its iterators or is destructed. This class template may only be used by deriving from it and passing the name of the derived class as its template parameter via the curiously recurring template pattern. The derived class must have `iterator` and `const_iterator` types that are instantiations of class template `_Safe_iterator` for this container and `local_iterator` and `const_local_iterator` types that are instantiations of class template `_Safe_local_iterator` for this container. Iterators will then be tracked automatically.

Definition at line 58 of file `safe_unordered_container.h`.

## 4.70.2 Member Function Documentation

4.70.2.1 `void __gnu_debug::_Safe_sequence_base::_M_attach ( _Safe_iterator_base * __it, bool __constant )`  
[*inherited*]

Attach an iterator to this sequence.

4.70.2.2 `void __gnu_debug::_Safe_unordered_container_base::_M_attach_local ( _Safe_iterator_base * __it, bool __constant )`  
[*inherited*]

Attach an iterator to this container.

4.70.2.3 `void __gnu_debug::_Safe_unordered_container_base::_M_attach_local_single ( _Safe_iterator_base * __it, bool __constant ) throw )` [*inherited*]

Likewise but not thread safe.

4.70.2.4 `void __gnu_debug::_Safe_sequence_base::_M_attach_single ( _Safe_iterator_base * __it, bool __constant ) throw )`  
[*inherited*]

Likewise but not thread safe.

4.70.2.5 `void __gnu_debug::_Safe_sequence_base::_M_detach ( _Safe_iterator_base * __it )` [*inherited*]

Detach an iterator from this sequence

4.70.2.6 `void __gnu_debug::_Safe_unordered_container_base::_M_detach_all ( )` [*protected*], [*inherited*]

Detach all iterators, leaving them singular.

4.70.2.7 `void __gnu_debug::Safe_unordered_container_base::M_detach_local ( _Safe_iterator_base * __it )`  
[inherited]

Detach an iterator from this container

4.70.2.8 `void __gnu_debug::Safe_unordered_container_base::M_detach_local_single ( _Safe_iterator_base * __it ) throw`  
[inherited]

Likewise but not thread safe.

4.70.2.9 `void __gnu_debug::Safe_sequence_base::M_detach_single ( _Safe_iterator_base * __it ) throw`  
[inherited]

Likewise but not thread safe.

4.70.2.10 `void __gnu_debug::Safe_sequence_base::M_detach_singular ( )` [protected], [inherited]

Detach all singular iterators.

Postcondition

for all iterators *i* attached to this sequence, *i*->\_M\_version == \_M\_version.

4.70.2.11 `__gnu_cxx::mutex& __gnu_debug::Safe_sequence_base::M_get_mutex ( ) throw` [protected],  
[inherited]

For use in \_Safe\_sequence.

4.70.2.12 `void __gnu_debug::Safe_sequence_base::M_invalidate_all ( ) const` [inline], [inherited]

Invalidates all iterators.

Definition at line 233 of file safe\_base.h.

4.70.2.13 `template<typename _Container > template<typename _Predicate > void __gnu_debug::Safe_unordered_container< _Container >::M_invalidate_if ( _Predicate __pred )`

Invalidates all iterators *x* that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.70.2.14 `template<typename _Container > template<typename _Predicate > void __gnu_debug::Safe_unordered_container< _Container >::M_invalidate_local_if ( _Predicate __pred )`

Invalidates all local iterators *x* that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal ilocal iterators nested in the safe ones.

4.70.2.15 `void __gnu_debug::Safe_sequence_base::M_revalidate_singular ( )` [protected], [inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.70.2.16 `void __gnu_debug::Safe_unordered_container_base::M_swap ( _Safe_unordered_container_base & __x )`  
[protected], [inherited]

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.70.2.17 `void __gnu_debug::_Safe_sequence_base::M_swap ( _Safe_sequence_base & __x )` [protected],  
[inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

### 4.70.3 Member Data Documentation

4.70.3.1 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::M_const_iterators` [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file `safe_base.h`.

4.70.3.2 `_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::M_const_local_iterators` [inherited]

The list of constant local iterators that reference this container.

Definition at line 131 of file `safe_unordered_base.h`.

4.70.3.3 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file `safe_base.h`.

4.70.3.4 `_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::M_local_iterators` [inherited]

The list of mutable local iterators that reference this container.

Definition at line 128 of file `safe_unordered_base.h`.

4.70.3.5 `unsigned int __gnu_debug::_Safe_sequence_base::M_version` [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 187 of file `safe_base.h`.

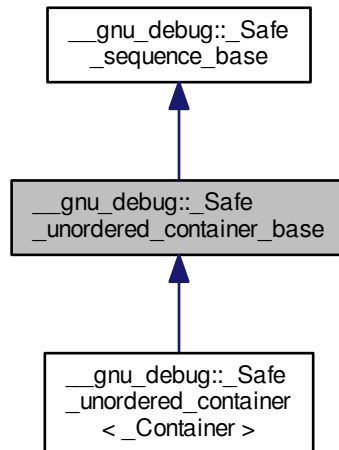
The documentation for this class was generated from the following file:

- [safe\\_unordered\\_container.h](#)



#### 4.71 `__gnu_debug::Safe_unordered_container_base` Class Reference

Inheritance diagram for `__gnu_debug::Safe_unordered_container_base`:



##### Public Member Functions

- `void _M_attach (_Safe_iterator_base * __it, bool __constant)`
- `void _M_attach_local (_Safe_iterator_base * __it, bool __constant)`
- `void _M_attach_local_single (_Safe_iterator_base * __it, bool __constant) throw ()`
- `void _M_attach_single (_Safe_iterator_base * __it, bool __constant) throw ()`
- `void _M_detach (_Safe_iterator_base * __it)`
- `void _M_detach_local (_Safe_iterator_base * __it)`
- `void _M_detach_local_single (_Safe_iterator_base * __it) throw ()`
- `void _M_detach_single (_Safe_iterator_base * __it) throw ()`
- `void _M_invalidate_all () const`

##### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_const_local_iterators`
- `_Safe_iterator_base * _M_iterators`
- `_Safe_iterator_base * _M_local_iterators`
- `unsigned int _M_version`

##### Protected Member Functions

- `_Safe_unordered_container_base (const _Safe_unordered_container_base &) noexcept`
- `_Safe_unordered_container_base (_Safe_unordered_container_base && __x) noexcept`

- `~_Safe_unordered_container_base ()`
- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_unordered_container_base &__x)`
- `void _M_swap (_Safe_sequence_base &__x)`

#### 4.71.1 Detailed Description

Base class that supports tracking of local iterators that reference an unordered container.

The `_Safe_unordered_container_base` class provides basic support for tracking iterators into an unordered container. Containers that track iterators must derived from `_Safe_unordered_container_base` publicly, so that safe iterators (which inherit `_Safe_iterator_base`) can attach to them. This class contains four linked lists of iterators, one for constant iterators, one for mutable iterators, one for constant local iterators, one for mutable local iterators and a version number that allows very fast invalidation of all iterators that reference the container.

This class must ensure that no operation on it may throw an exception, otherwise *safe* containers may fail to provide the exception-safety guarantees required by the C++ standard.

Definition at line 123 of file `safe_unordered_base.h`.

#### 4.71.2 Constructor & Destructor Documentation

4.71.2.1 `__gnu_debug::_Safe_unordered_container_base::~~_Safe_unordered_container_base ( ) [inline], [protected]`

Notify all iterators that reference this container that the container is being destroyed.

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

#### 4.71.3 Member Function Documentation

4.71.3.1 `void __gnu_debug::_Safe_sequence_base::_M_attach ( _Safe_iterator_base * __it, bool __constant ) [inherited]`

Attach an iterator to this sequence.

4.71.3.2 `void __gnu_debug::_Safe_unordered_container_base::_M_attach_local ( _Safe_iterator_base * __it, bool __constant )`

Attach an iterator to this container.

4.71.3.3 `void __gnu_debug::_Safe_unordered_container_base::_M_attach_local_single ( _Safe_iterator_base * __it, bool __constant ) throw )`

Likewise but not thread safe.

4.71.3.4 `void __gnu_debug::_Safe_sequence_base::_M_attach_single ( _Safe_iterator_base * __it, bool __constant ) throw ) [inherited]`

Likewise but not thread safe.

4.71.3.5 void \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_detach ( \_Safe\_iterator\_base \* \_\_it ) [inherited]

Detach an iterator from this sequence

4.71.3.6 void \_\_gnu\_debug::\_Safe\_unordered\_container\_base::\_M\_detach\_all ( ) [protected]

Detach all iterators, leaving them singular.

4.71.3.7 void \_\_gnu\_debug::\_Safe\_unordered\_container\_base::\_M\_detach\_local ( \_Safe\_iterator\_base \* \_\_it )

Detach an iterator from this container

4.71.3.8 void \_\_gnu\_debug::\_Safe\_unordered\_container\_base::\_M\_detach\_local\_single ( \_Safe\_iterator\_base \* \_\_it ) throw )

Likewise but not thread safe.

4.71.3.9 void \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_detach\_single ( \_Safe\_iterator\_base \* \_\_it ) throw )  
[inherited]

Likewise but not thread safe.

4.71.3.10 void \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_detach\_singular ( ) [protected],[inherited]

Detach all singular iterators.

#### Postcondition

for all iterators i attached to this sequence, i->\_M\_version == \_M\_version.

4.71.3.11 \_\_gnu\_cxx::mutex& \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_get\_mutex ( ) throw ) [protected],  
[inherited]

For use in \_Safe\_sequence.

4.71.3.12 void \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_invalidate\_all ( ) const [inline],[inherited]

Invalidates all iterators.

Definition at line 233 of file safe\_base.h.

4.71.3.13 void \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_revalidate\_singular ( ) [protected],[inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.71.3.14 void \_\_gnu\_debug::\_Safe\_unordered\_container\_base::\_M\_swap ( \_Safe\_unordered\_container\_base & \_\_x )  
[protected]

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.71.3.15 void \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_swap ( \_Safe\_sequence\_base & \_\_x ) [protected],  
[inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

## 4.71.4 Member Data Documentation

4.71.4.1 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_const_iterators` [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file `safe_base.h`.

4.71.4.2 `_Safe_iterator_base* __gnu_debug::Safe_unordered_container_base::M_const_local_iterators`

The list of constant local iterators that reference this container.

Definition at line 131 of file `safe_unordered_base.h`.

4.71.4.3 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file `safe_base.h`.

4.71.4.4 `_Safe_iterator_base* __gnu_debug::Safe_unordered_container_base::M_local_iterators`

The list of mutable local iterators that reference this container.

Definition at line 128 of file `safe_unordered_base.h`.

4.71.4.5 `unsigned int __gnu_debug::Safe_sequence_base::M_version` [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 187 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [safe\\_unordered\\_base.h](#)

4.72 `__gnu_parallel::__accumulate_binop_reduct<_BinOp>` Struct Template Reference

## Public Member Functions

- `__accumulate_binop_reduct` (`_BinOp &__b`)
- `template<typename _Result, typename _Addend> _Result operator()` (`const _Result &__x, const _Addend &__y`)

## Public Attributes

- `_BinOp &__binop`

## 4.72.1 Detailed Description

```
template<typename _BinOp>struct __gnu_parallel::__accumulate_binop_reduct<_BinOp>
```

General reduction, using a binary operator.

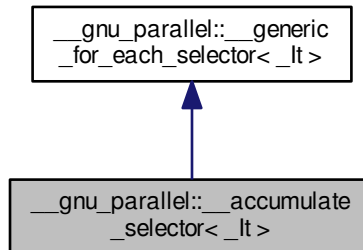
Definition at line 335 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

### 4.73 `__gnu_parallel::__accumulate_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__accumulate_selector<_It>`:



#### Public Member Functions

- `template<typename _Op> std::iterator_traits<_It>::value_type operator() (_Op __o, _It __i)`

#### Public Attributes

- `_It _M_finish_iterator`

#### 4.73.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__accumulate_selector<_It>`

`std::accumulate()` selector.

Definition at line 208 of file `for_each_selectors.h`.

#### 4.73.2 Member Function Documentation

4.73.2.1 `template<typename _It> template<typename _Op> std::iterator_traits<_It>::value_type __gnu_parallel::__accumulate_selector<_It>::operator() (_Op __o, _It __i) [inline]`

Functor execution.

##### Parameters

<code>__o</code>	Operator (unused).
<code>__i</code>	iterator referencing object.

##### Returns

The current value.

Definition at line 216 of file `for_each_selectors.h`.

## 4.73.3 Member Data Documentation

4.73.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::M_finish_iterator`  
[inherited]

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

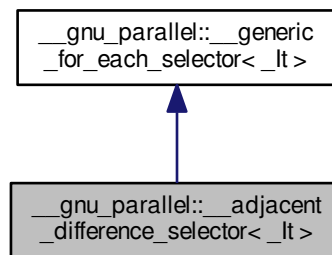
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

4.74 `__gnu_parallel::__adjacent_difference_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__adjacent_difference_selector<_It>`:



## Public Member Functions

- `template<typename _Op> bool operator() (_Op &__o, _It __i)`

## Public Attributes

- `_It M\_finish\_iterator`

## 4.74.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__adjacent_difference_selector<_It>`

Selector that returns the difference between two adjacent `__elements`.

Definition at line 269 of file `for_each_selectors.h`.

## 4.74.2 Member Data Documentation

#### 4.74.2.1 `template<typename _It > _It __gnu_parallel::__generic_for_each_selector< _It >::M_finish_iterator` [inherited]

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

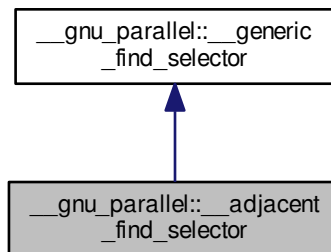
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

### 4.75 `__gnu_parallel::__adjacent_find_selector` Struct Reference

Inheritance diagram for `__gnu_parallel::__adjacent_find_selector`:



#### Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred > std::pair< _RAIter1, _RAIter2 > \_M\_sequential\_algorithm ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred > bool operator\(\) ( _RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

#### 4.75.1 Detailed Description

Test predicate on two adjacent elements.

Definition at line 80 of file `find_selectors.h`.

#### 4.75.2 Member Function Documentation

##### 4.75.2.1 `template<typename _RAIter1, typename _RAIter2, typename _Pred > std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__adjacent_find_selector::M_sequential_algorithm ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred )` [inline]

Corresponding sequential algorithm on a sequence.

## Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__pred</code>	Find predicate.

Definition at line 105 of file `find_selectors.h`.

References `std::make_pair()`.

4.75.2.2 `template<typename _RAIter1, typename _RAIter2, typename _Pred> bool __gnu_parallel::__adjacent_find_selector(`  
`::operator() ( _RAIter1 __i1, _RAIter2 __i2, _Pred __pred )` `[inline]`

Test on one position.

## Parameters

<code>__i1</code>	_literator on first sequence.
<code>__i2</code>	_literator on second sequence (unused).
<code>__pred</code>	Find predicate.

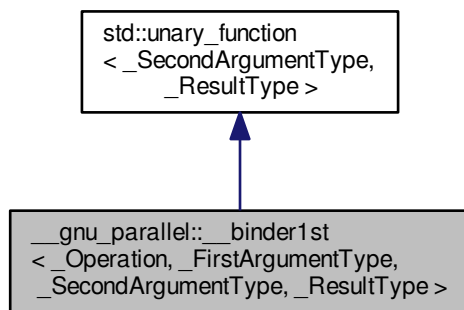
Definition at line 90 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

4.76 `__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`  
Class Template Reference

Inheritance diagram for `__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`:



## Public Types

- typedef `_SecondArgumentType` [argument\\_type](#)
- typedef `_ResultType` [result\\_type](#)



## Public Member Functions

- **\_\_binder1st** (const \_Operation &\_\_x, const \_FirstArgumentType &\_\_y)
- \_ResultType **operator()** (const \_SecondArgumentType &\_\_x)
- \_ResultType **operator()** (\_SecondArgumentType &\_\_x) const

## Protected Attributes

- \_Operation **\_M\_op**
- \_FirstArgumentType **\_M\_value**

### 4.76.1 Detailed Description

```
template<typename _Operation, typename _FirstArgumentType, typename _SecondArgumentType, typename _ResultType>class ↵
__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >
```

Similar to std::binder1st, but giving the argument types explicitly.

Definition at line 192 of file parallel/base.h.

### 4.76.2 Member Typedef Documentation

**4.76.2.1** `typedef _SecondArgumentType std::unary_function< _SecondArgumentType , _ResultType >::argument_type`  
[inherited]

argument\_type is the type of the argument

Definition at line 104 of file stl\_function.h.

**4.76.2.2** `typedef _ResultType std::unary_function< _SecondArgumentType , _ResultType >::result_type` [inherited]

result\_type is the return type

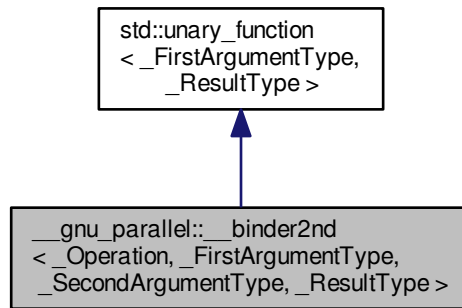
Definition at line 107 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

## 4.77 `__gnu_parallel::__binder2nd<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType>` Class Template Reference

Inheritance diagram for `__gnu_parallel::__binder2nd<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType>`:



### Public Types

- typedef `_FirstArgumentType` [argument\\_type](#)
- typedef `_ResultType` [result\\_type](#)

### Public Member Functions

- **`__binder2nd`** (`const _Operation &__x`, `const _SecondArgumentType &__y`)
- `_ResultType` **`operator()`** (`const _FirstArgumentType &__x`) `const`
- `_ResultType` **`operator()`** (`_FirstArgumentType &__x`)

### Protected Attributes

- `_Operation` **`_M_op`**
- `_SecondArgumentType` **`_M_value`**

#### 4.77.1 Detailed Description

`template<typename _Operation, typename _FirstArgumentType, typename _SecondArgumentType, typename _ResultType> class __gnu_parallel::__binder2nd<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType>`

Similar to `std::binder2nd`, but giving the argument types explicitly.

Definition at line 220 of file `parallel/base.h`.

#### 4.77.2 Member Typedef Documentation

##### 4.77.2.1 `typedef _FirstArgumentType std::unary_function< _FirstArgumentType , _ResultType >::argument_type` [inherited]

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

##### 4.77.2.2 `typedef _ResultType std::unary_function< _FirstArgumentType , _ResultType >::result_type` [inherited]

`result_type` is the return type

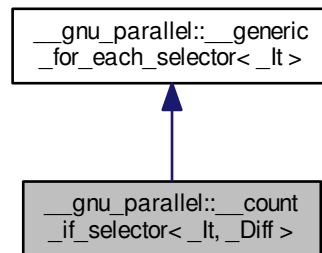
Definition at line 107 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

#### 4.78 `__gnu_parallel::__count_if_selector< _It, _Diff >` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__count_if_selector< _It, _Diff >`:



#### Public Member Functions

- `template<typename _Op > _Diff operator() (_Op &__o, _It __i)`

#### Public Attributes

- `_It _M_finish_iterator`

#### 4.78.1 Detailed Description

`template<typename _It, typename _Diff> struct __gnu_parallel::__count_if_selector< _It, _Diff >`

`std::count_if ()` selector.

Definition at line 194 of file `for_each_selectors.h`.

## 4.78.2 Member Function Documentation

4.78.2.1 `template<typename _It, typename _Diff> template<typename _Op> _Diff __gnu_parallel::__count_if_selector<_It, _Diff>::operator()( _Op &__o, _It __i ) [inline]`

Functor execution.

Parameters

<code>__o</code>	Operator.
<code>__i</code>	iterator referencing object.

Returns

1 if count, 0 if does not count.

Definition at line 202 of file `for_each_selectors.h`.

## 4.78.3 Member Data Documentation

4.78.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::M_finish_iterator [inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

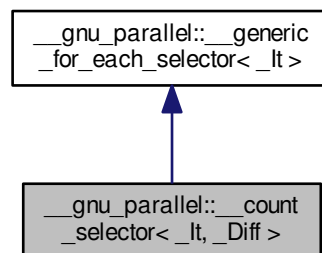
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 4.79 \_\_gnu\_parallel::\_\_count\_selector&lt;\_It, \_Diff&gt; Struct Template Reference

Inheritance diagram for `__gnu_parallel::__count_selector<_It, _Diff>`:



## Public Member Functions

- `template<typename _ValueType> _Diff operator() ( _ValueType &__v, _It __i)`

## Public Attributes

- [\\_lt \\_M\\_finish\\_iterator](#)

### 4.79.1 Detailed Description

template<typename \_It, typename \_Diff> struct \_\_gnu\_parallel::\_\_count\_selector< \_It, \_Diff >

std::count() selector.

Definition at line 180 of file for\_each\_selectors.h.

### 4.79.2 Member Function Documentation

4.79.2.1 template<typename \_It, typename \_Diff> template<typename \_ValueType > \_Diff \_\_gnu\_parallel::\_\_count\_selector< \_It, \_Diff >::operator() ( \_ValueType & \_\_v, \_It \_\_i ) [inline]

Functor execution.

#### Parameters

<code>__v</code>	Current value.
<code>__i</code>	iterator referencing object.

#### Returns

1 if count, 0 if does not count.

Definition at line 188 of file for\_each\_selectors.h.

### 4.79.3 Member Data Documentation

4.79.3.1 template<typename \_It > \_It \_\_gnu\_parallel::\_\_generic\_for\_each\_selector< \_It >::\_\_M\_finish\_iterator [inherited]

\_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).

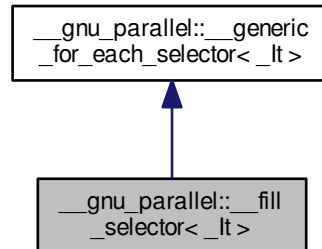
Definition at line 47 of file for\_each\_selectors.h.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

4.80 `__gnu_parallel::__fill_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__fill_selector<_It>`:



## Public Member Functions

- `template<typename _ValueType> bool operator() (_ValueType &__v, _It __i)`

## Public Attributes

- `_It _M_finish_iterator`

## 4.80.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__fill_selector<_It>`

`std::fill()` selector.

Definition at line 84 of file `for_each_selectors.h`.

## 4.80.2 Member Function Documentation

4.80.2.1 `template<typename _It> template<typename _ValueType> bool __gnu_parallel::__fill_selector<_It>::operator() (_ValueType &__v, _It __i) [inline]`

Functor execution.

## Parameters

<code>__v</code>	Current value.
<code>__i</code>	iterator referencing object.

Definition at line 91 of file `for_each_selectors.h`.

## 4.80.3 Member Data Documentation

4.80.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator`  
`[inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

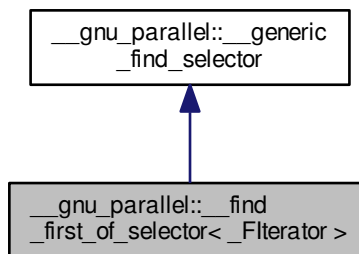
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

#### 4.81 `__gnu_parallel::__find_first_of_selector<_FIterator>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__find_first_of_selector<_FIterator>`:



##### Public Member Functions

- `__find_first_of_selector` (`_FIterator __begin`, `_FIterator __end`)
- `template<typename _RAIter1, typename _RAIter2, typename _Pred> std::pair<_RAIter1, _RAIter2> __M_sequential_algorithm` (`_RAIter1 __begin1`, `_RAIter1 __end1`, `_RAIter2 __begin2`, `_Pred __pred`)
- `template<typename _RAIter1, typename _RAIter2, typename _Pred> bool operator()` (`_RAIter1 __i1`, `_RAIter2 __i2`, `_Pred __pred`)

##### Public Attributes

- `_FIterator __M_begin`
- `_FIterator __M_end`

##### 4.81.1 Detailed Description

`template<typename _FIterator> struct __gnu_parallel::__find_first_of_selector<_FIterator>`

Test predicate on several elements.

Definition at line 153 of file `find_selectors.h`.

## 4.81.2 Member Function Documentation

4.81.2.1 `template<typename _FIterator> template<typename _RAIter1, typename _RAIter2, typename _Pred> std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_first_of_selector<_FIterator>::__M_sequential_algorithm ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred ) [inline]`

Corresponding sequential algorithm on a sequence.

## Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__pred</code>	Find predicate.

Definition at line 186 of file `find_selectors.h`.

References `std::make_pair()`.

4.81.2.2 `template<typename _FIterator> template<typename _RAIter1, typename _RAIter2, typename _Pred> bool __gnu_parallel::__find_first_of_selector<_FIterator>::operator() ( _RAIter1 __i1, _RAIter2 __i2, _Pred __pred ) [inline]`

Test on one position.

## Parameters

<code>__i1</code>	_Iterator on first sequence.
<code>__i2</code>	_Iterator on second sequence (unused).
<code>__pred</code>	Find predicate.

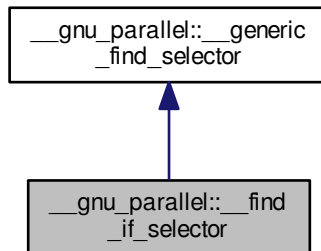
Definition at line 169 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

## 4.82 \_\_gnu\_parallel::\_\_find\_if\_selector Struct Reference

Inheritance diagram for `__gnu_parallel::__find_if_selector`:





## Public Member Functions

- `template<typename _RAIter1 , typename _RAIter2 , typename _Pred > std::pair< _RAIter1, _RAIter2 > \_M\_sequential\_algorithm ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1 , typename _RAIter2 , typename _Pred > bool operator() ( _RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

### 4.82.1 Detailed Description

Test predicate on a single element, used for `std::find()` and `std::find_if()`.

Definition at line 50 of file `find_selectors.h`.

### 4.82.2 Member Function Documentation

**4.82.2.1** `template<typename _RAIter1 , typename _RAIter2 , typename _Pred > std::pair< _RAIter1, _RAIter2 > \_\_gnu\_parallel::\_\_find\_if\_selector::\_\_M\_sequential\_algorithm ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred ) \[inline\]`

Corresponding sequential algorithm on a sequence.

#### Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__pred</code>	Find predicate.

Definition at line 72 of file `find_selectors.h`.

References `std::make_pair()`.

**4.82.2.2** `template<typename _RAIter1 , typename _RAIter2 , typename _Pred > bool \_\_gnu\_parallel::\_\_find\_if\_selector::operator() ( _RAIter1 __i1, _RAIter2 __i2, _Pred __pred ) \[inline\]`

Test on one position.

#### Parameters

<code>__i1</code>	Iterator on first sequence.
<code>__i2</code>	Iterator on second sequence (unused).
<code>__pred</code>	Find predicate.

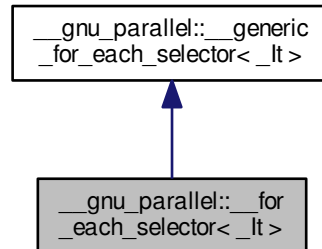
Definition at line 60 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

4.83 `__gnu_parallel::__for_each_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__for_each_selector<_It>`:



## Public Member Functions

- `template<typename _Op> bool operator() (_Op &__o, _It __i)`

## Public Attributes

- `_It _M_finish_iterator`

## 4.83.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__for_each_selector<_It>`

`std::for_each()` selector.

Definition at line 52 of file `for_each_selectors.h`.

## 4.83.2 Member Function Documentation

4.83.2.1 `template<typename _It> template<typename _Op> bool __gnu_parallel::__for_each_selector<_It>::operator() (_Op &__o, _It __i) [inline]`

Functor execution.

## Parameters

<code>__o</code>	Operator.
<code>__i</code>	iterator referencing object.

Definition at line 59 of file `for_each_selectors.h`.

## 4.83.3 Member Data Documentation

#### 4.83.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator` [inherited]

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

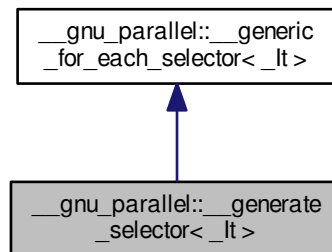
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

### 4.84 `__gnu_parallel::__generate_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__generate_selector<_It>`:



#### Public Member Functions

- `template<typename _Op> bool operator() (_Op &__o, _It __i)`

#### Public Attributes

- `_It __M_finish_iterator`

#### 4.84.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__generate_selector<_It>`

`std::generate()` selector.

Definition at line 68 of file `for_each_selectors.h`.

#### 4.84.2 Member Function Documentation

4.84.2.1 `template<typename _It> template<typename _Op> bool __gnu_parallel::__generate_selector<_It>::operator()  
( _Op & __o, _It __i ) [inline]`

Functor execution.

## Parameters

<code>__o</code>	Operator.
<code>__i</code>	iterator referencing object.

Definition at line 75 of file `for_each_selectors.h`.

## 4.84.3 Member Data Documentation

#### 4.84.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator` [inherited]

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

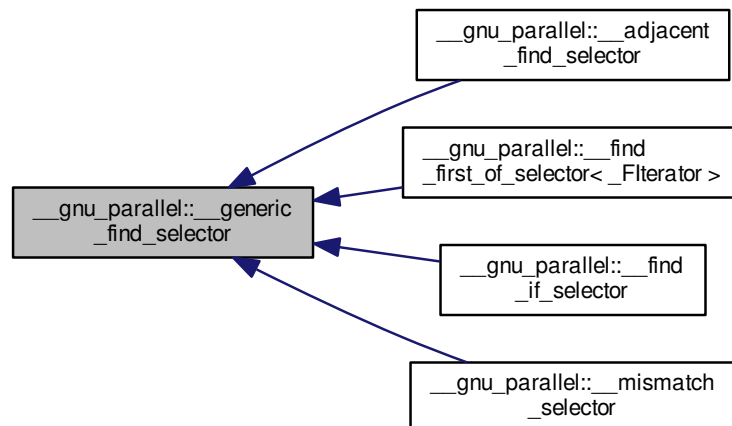
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

### 4.85 `__gnu_parallel::__generic_find_selector` Struct Reference

Inheritance diagram for `__gnu_parallel::__generic_find_selector`:



## 4.85.1 Detailed Description

Base class of all `__gnu_parallel::__find_template` selectors.

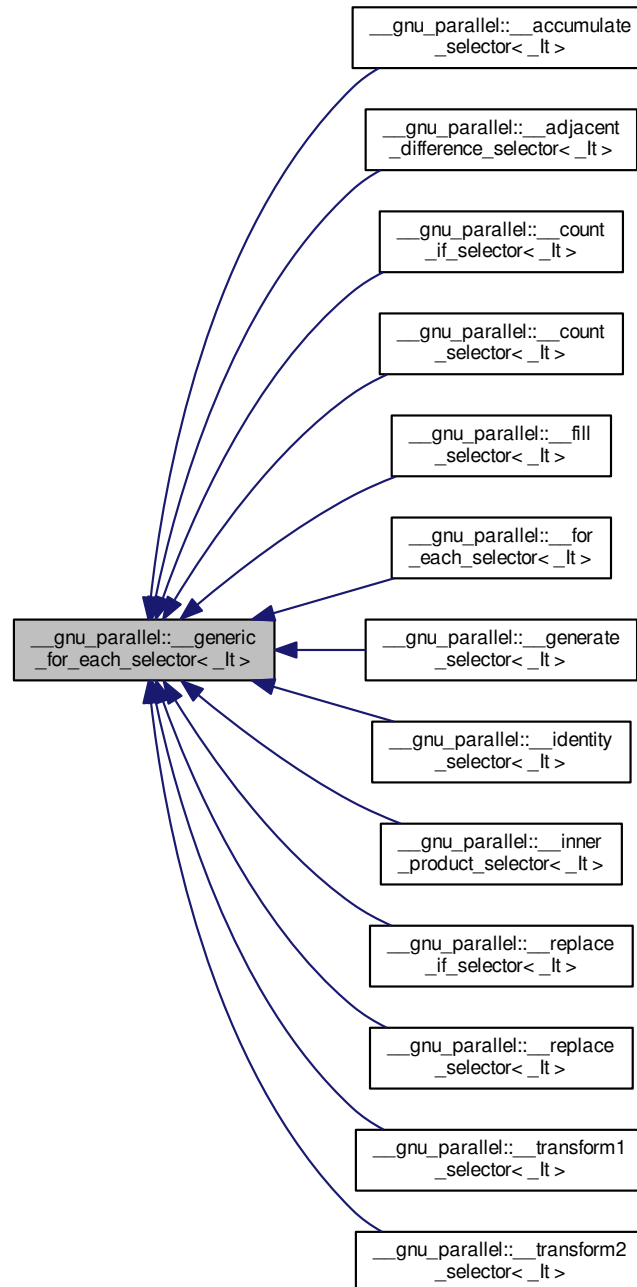
Definition at line 43 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

## 4.86 \_\_gnu\_parallel::\_\_generic\_for\_each\_selector&lt;\_It&gt; Struct Template Reference

Inheritance diagram for \_\_gnu\_parallel::\_\_generic\_for\_each\_selector<\_It>:



## Public Attributes

- [\\_It \\_M\\_finish\\_iterator](#)

### 4.86.1 Detailed Description

template<typename \_It>struct \_\_gnu\_parallel::\_\_generic\_for\_each\_selector<\_It>

Generic \_\_selector for embarrassingly parallel functions.

Definition at line 42 of file for\_each\_selectors.h.

### 4.86.2 Member Data Documentation

4.86.2.1 template<typename \_It>\_It \_\_gnu\_parallel::\_\_generic\_for\_each\_selector<\_It>::\_\_M\_finish\_iterator

\_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).

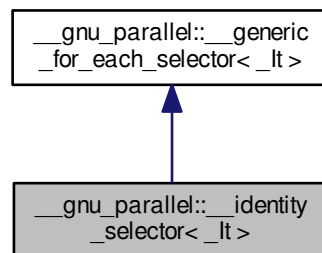
Definition at line 47 of file for\_each\_selectors.h.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 4.87 \_\_gnu\_parallel::\_\_identity\_selector<\_It> Struct Template Reference

Inheritance diagram for \_\_gnu\_parallel::\_\_identity\_selector<\_It>:



## Public Member Functions

- template<typename \_Op>\_It [operator\(\)](#) (\_Op \_\_o, \_It \_\_i)

## Public Attributes

- [\\_It \\_M\\_finish\\_iterator](#)

## 4.87.1 Detailed Description

```
template<typename _It>struct __gnu_parallel::__identity_selector<_It>
```

Selector that just returns the passed iterator.

Definition at line 253 of file `for_each_selectors.h`.

## 4.87.2 Member Function Documentation

4.87.2.1 `template<typename _It> template<typename _Op> _It __gnu_parallel::__identity_selector<_It>::operator() ( _Op __o, _It __i ) [inline]`

Functor execution.

## Parameters

<code>__o</code>	Operator (unused).
<code>__i</code>	iterator referencing object.

## Returns

Passed iterator.

Definition at line 261 of file `for_each_selectors.h`.

## 4.87.3 Member Data Documentation

4.87.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::M_finish_iterator [inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

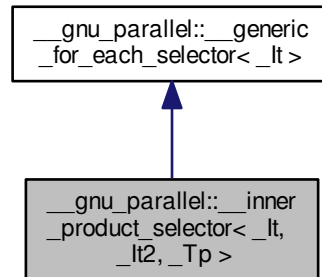
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)



#### 4.88 `__gnu_parallel::__inner_product_selector<_It,_It2,_Tp>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__inner_product_selector<_It,_It2,_Tp>`:



##### Public Member Functions

- [`\_\_inner\_product\_selector`](#) (`_It __b1, _It2 __b2`)
- `template<typename _Op> _Tp operator\(\) (_Op __mult, _It __current)`

##### Public Attributes

- `_It \_\_begin1\_iterator`
- `_It2 \_\_begin2\_iterator`
- `_It \_\_M\_finish\_iterator`

##### 4.88.1 Detailed Description

`template<typename _It, typename _It2, typename _Tp> struct __gnu_parallel::__inner_product_selector<_It,_It2,_Tp>`

`std::inner_product()` selector.

Definition at line 222 of file `for_each_selectors.h`.

##### 4.88.2 Constructor & Destructor Documentation

4.88.2.1 `template<typename _It, typename _It2, typename _Tp> __gnu_parallel::__inner_product_selector<_It,_It2,_Tp>::__inner_product_selector(_It __b1, _It2 __b2) [inline],[explicit]`

Constructor.

## Parameters

<code>__b1</code>	Begin iterator of first sequence.
<code>__b2</code>	Begin iterator of second sequence.

Definition at line 234 of file `for_each_selectors.h`.

## 4.88.3 Member Function Documentation

4.88.3.1 `template<typename _It, typename _It2, typename _Tp> template<typename _Op> _Tp  
__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::operator() ( _Op __mult, _It __current )  
[inline]`

Functor execution.

## Parameters

<code>__mult</code>	Multiplication functor.
<code>__current</code>	iterator referencing object.

## Returns

Inner product elemental `__result`.

Definition at line 243 of file `for_each_selectors.h`.

References `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::__begin1_iterator`.

## 4.88.4 Member Data Documentation

4.88.4.1 `template<typename _It, typename _It2, typename _Tp> _It __gnu_parallel::__inner_product_selector<_It, _It2,  
_Tp>::__begin1_iterator`

Begin iterator of first sequence.

Definition at line 225 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::operator()()`.

4.88.4.2 `template<typename _It, typename _It2, typename _Tp> _It2 __gnu_parallel::__inner_product_selector<_It, _It2,  
_Tp>::__begin2_iterator`

Begin iterator of second sequence.

Definition at line 228 of file `for_each_selectors.h`.

4.88.4.3 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator  
[inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

#### 4.89 `__gnu_parallel::__max_element_reduct<_Compare, _It>` Struct Template Reference

##### Public Member Functions

- `__max_element_reduct` (`_Compare &__c`)
- `_It operator()` (`_It __x, _It __y`)

##### Public Attributes

- `_Compare & __comp`

##### 4.89.1 Detailed Description

`template<typename _Compare, typename _It>struct __gnu_parallel::__max_element_reduct<_Compare, _It>`

Reduction for finding the maximum element, using a comparator.

Definition at line 321 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

#### 4.90 `__gnu_parallel::__min_element_reduct<_Compare, _It>` Struct Template Reference

##### Public Member Functions

- `__min_element_reduct` (`_Compare &__c`)
- `_It operator()` (`_It __x, _It __y`)

##### Public Attributes

- `_Compare & __comp`

##### 4.90.1 Detailed Description

`template<typename _Compare, typename _It>struct __gnu_parallel::__min_element_reduct<_Compare, _It>`

Reduction for finding the maximum element, using a comparator.

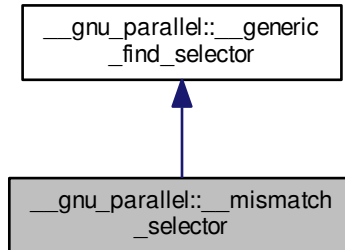
Definition at line 307 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 4.91 \_\_gnu\_parallel::\_\_mismatch\_selector Struct Reference

Inheritance diagram for \_\_gnu\_parallel::\_\_mismatch\_selector:



## Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred > std::pair<_RAIter1, _RAIter2 > \_M\_sequential\_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred > bool operator\(\) (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

## 4.91.1 Detailed Description

Test inverted predicate on a single element.

Definition at line 119 of file `find_selectors.h`.

## 4.91.2 Member Function Documentation

4.91.2.1 `template<typename _RAIter1, typename _RAIter2, typename _Pred > std::pair<_RAIter1, _RAIter2> __gnu_parallel::__mismatch_selector::_M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred) [inline]`

Corresponding sequential algorithm on a sequence.

Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__pred</code>	Find predicate.

Definition at line 143 of file `find_selectors.h`.

4.91.2.2 `template<typename _RAIter1, typename _RAIter2, typename _Pred > bool __gnu_parallel::__mismatch_selector::operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred) [inline]`

Test on one position.

## Parameters

<code>__i1</code>	Iterator on first sequence.
<code>__i2</code>	Iterator on second sequence (unused).
<code>__pred</code>	Find predicate.

Definition at line 130 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

#### 4.92 `__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterliterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

## Public Member Functions

- `_RAIter3 operator() ( _RAIterliterator __seqs_begin, _RAIterliterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

##### 4.92.1 Detailed Description

```
template<bool __sentinels, typename _RAIterliterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterliterator, _RAIter3, _DifferenceTp, _Compare >
```

Switch for 3-way merging with `__sentinels` turned off.

Note that 3-way merging is always stable!

Definition at line 752 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

#### 4.93 `__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterliterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

## Public Member Functions

- `_RAIter3 operator() ( _RAIterliterator __seqs_begin, _RAIterliterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

##### 4.93.1 Detailed Description

```
template<typename _RAIterliterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterliterator, _RAIter3, _DifferenceTp, _Compare >
```

Switch for 3-way merging with `__sentinels` turned on.

Note that 3-way merging is always stable!

Definition at line 772 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

#### 4.94 `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

##### Public Member Functions

- `_RAIter3 operator()` (`_RAIterlterator __seqs_begin`, `_RAIterlterator __seqs_end`, `_RAIter3 __target`, `_DifferenceTp __length`, `_Compare __comp`)

##### 4.94.1 Detailed Description

`template<bool __sentinels, typename _RAIterlterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare >`

Switch for 4-way merging with `__sentinels` turned off.

Note that 4-way merging is always stable!

Definition at line 795 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

#### 4.95 `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

##### Public Member Functions

- `_RAIter3 operator()` (`_RAIterlterator __seqs_begin`, `_RAIterlterator __seqs_end`, `_RAIter3 __target`, `_DifferenceTp __length`, `_Compare __comp`)

##### 4.95.1 Detailed Description

`template<typename _RAIterlterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare >`

Switch for 4-way merging with `__sentinels` turned on.

Note that 4-way merging is always stable!

Definition at line 815 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

#### 4.96 `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

## Public Member Functions

- **\_RAIter3 operator()** (\_RAIterIterator \_\_seqs\_begin, \_RAIterIterator \_\_seqs\_end, \_RAIter3 \_\_target, const typename std::iterator\_traits< typename std::iterator\_traits< \_RAIterIterator >::value\_type::first\_type >::value\_type & \_\_sentinel, \_DifferenceTp \_\_length, \_Compare \_\_comp)

## 4.96.1 Detailed Description

```
template<bool __sentinels, bool __stable, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare> struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
```

Switch for k-way merging with \_\_sentinels turned on.

Definition at line 837 of file multiway\_merge.h.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

#### 4.97 \_\_gnu\_parallel::\_\_multiway\_merge\_k\_variant\_sentinel\_switch< false, \_\_stable, \_RAIterIterator, \_RAIter3, \_DifferenceTp, \_Compare > Struct Template Reference

## Public Member Functions

- **\_RAIter3 operator()** (\_RAIterIterator \_\_seqs\_begin, \_RAIterIterator \_\_seqs\_end, \_RAIter3 \_\_target, const typename std::iterator\_traits< typename std::iterator\_traits< \_RAIterIterator >::value\_type::first\_type >::value\_type & \_\_sentinel, \_DifferenceTp \_\_length, \_Compare \_\_comp)

## 4.97.1 Detailed Description

```
template<bool __stable, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare> struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
```

Switch for k-way merging with \_\_sentinels turned off.

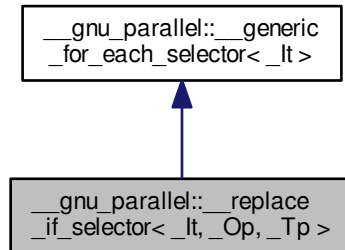
Definition at line 872 of file multiway\_merge.h.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

4.98 `__gnu_parallel::__replace_if_selector<_It,_Op,_Tp>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__replace_if_selector<_It,_Op,_Tp>`:



## Public Member Functions

- `__replace_if_selector` (const `_Tp` & `__new_val`)
- bool `operator()` (`_Op` & `__o`, `_It` `__i`)

## Public Attributes

- const `_Tp` & `__new_val`
- `_It` `_M_finish_iterator`

## 4.98.1 Detailed Description

`template<typename _It, typename _Op, typename _Tp> struct __gnu_parallel::__replace_if_selector<_It,_Op,_Tp>`

`std::replace()` selector.

Definition at line 156 of file `for_each_selectors.h`.

## 4.98.2 Constructor &amp; Destructor Documentation

4.98.2.1 `template<typename _It, typename _Op, typename _Tp> __gnu_parallel::__replace_if_selector<_It,_Op,_Tp>::__replace_if_selector( const _Tp & __new_val ) [inline],[explicit]`

Constructor.

## Parameters

<code>__new_val</code>	Value to replace with.
------------------------	------------------------

Definition at line 164 of file `for_each_selectors.h`.



#### 4.98.3 Member Function Documentation

4.98.3.1 `template<typename _It, typename _Op, typename _Tp> bool __gnu_parallel::__replace_if_selector<_It, _Op, _Tp>::operator() ( _Op & __o, _It __i ) [inline]`

Functor execution.

## Parameters

<code>__o</code>	Operator.
<code>__i</code>	iterator referencing object.

Definition at line 170 of file `for_each_selectors.h`.

References `__gnu_parallel::__replace_if_selector<_It,_Op,_Tp>::__new_val`.

## 4.98.4 Member Data Documentation

4.98.4.1 `template<typename _It, typename _Op, typename _Tp> const _Tp& __gnu_parallel::__replace_if_selector<_It, _Op, _Tp>::__new_val`

Value to replace with.

Definition at line 159 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__replace_if_selector<_It,_Op,_Tp>::operator()`.

4.98.4.2 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator`  
[inherited]

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

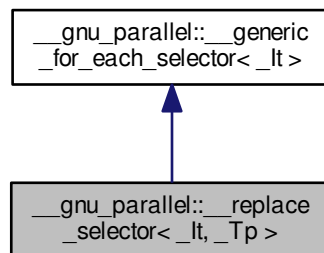
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 4.99 \_\_gnu\_parallel::\_\_replace\_selector&lt;\_It,\_Tp&gt; Struct Template Reference

Inheritance diagram for `__gnu_parallel::__replace_selector<_It,_Tp>`:



## Public Member Functions

- `__replace_selector` (`const _Tp &__new_val`)
- `bool operator()` (`_Tp &__v, _It __i`)

## Public Attributes

- `const _Tp & __new_val`
- `_It __M_finish_iterator`

## 4.99.1 Detailed Description

`template<typename _It, typename _Tp> struct __gnu_parallel::__replace_selector<_It, _Tp>`

`std::replace()` selector.

Definition at line 132 of file `for_each_selectors.h`.

## 4.99.2 Constructor &amp; Destructor Documentation

4.99.2.1 `template<typename _It, typename _Tp> __gnu_parallel::__replace_selector<_It, _Tp>::__replace_selector ( const _Tp & __new_val ) [inline], [explicit]`

Constructor.

## Parameters

<code>__new_val</code>	Value to replace with.
------------------------	------------------------

Definition at line 140 of file `for_each_selectors.h`.

## 4.99.3 Member Function Documentation

4.99.3.1 `template<typename _It, typename _Tp> bool __gnu_parallel::__replace_selector<_It, _Tp>::operator() ( _Tp & __v, _It __i ) [inline]`

Functor execution.

## Parameters

<code>__v</code>	Current value.
<code>__i</code>	iterator referencing object.

Definition at line 146 of file `for_each_selectors.h`.

References `__gnu_parallel::__replace_selector<_It, _Tp>::__new_val`.

## 4.99.4 Member Data Documentation

4.99.4.1 `template<typename _It, typename _Tp> const _Tp& __gnu_parallel::__replace_selector<_It, _Tp>::__new_val`

Value to replace with.

Definition at line 135 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__replace_selector<_It, _Tp>::operator()()`.

4.99.4.2 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator [inherited]`

`_It` iterator on last element processed; needed for some algorithms (e. g. `std::transform()`).

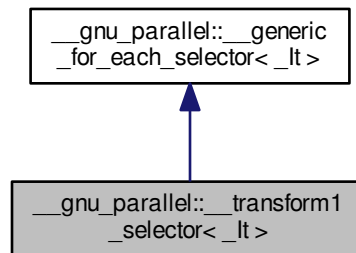
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 4.100 `__gnu_parallel::__transform1_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__transform1_selector<_It>`:



### Public Member Functions

- `template<typename _Op> bool operator() (_Op &__o, _It __i)`

### Public Attributes

- `_It _M_finish_iterator`

#### 4.100.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__transform1_selector<_It>`

`std::transform()` `__selector`, one input sequence variant.

Definition at line 100 of file `for_each_selectors.h`.

#### 4.100.2 Member Function Documentation

4.100.2.1 `template<typename _It> template<typename _Op> bool __gnu_parallel::__transform1_selector<_It>::operator() (_Op &__o, _It __i) [inline]`

Functor execution.

**Parameters**

<code>__o</code>	Operator.
<code>__i</code>	iterator referencing object.

Definition at line 107 of file `for_each_selectors.h`.

**4.100.3 Member Data Documentation**
**4.100.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator`**  
 [inherited]

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

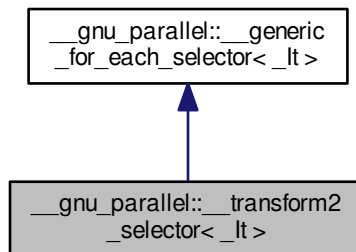
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

**4.101 `__gnu_parallel::__transform2_selector<_It>` Struct Template Reference**

Inheritance diagram for `__gnu_parallel::__transform2_selector<_It>`:

**Public Member Functions**

- `template<typename _Op> bool operator() (_Op &__o, _It __i)`

**Public Attributes**

- `_It __M_finish_iterator`

**4.101.1 Detailed Description**

```
template<typename _It>struct __gnu_parallel::__transform2_selector<_It>
```

`std::transform()` \_\_selector, two input sequences variant.

Definition at line 116 of file `for_each_selectors.h`.

#### 4.101.2 Member Function Documentation

4.101.2.1 `template<typename _It> template<typename _Op> bool __gnu_parallel::__transform2_selector<_It>::operator()( _Op &__o, _It __i ) [inline]`

Functor execution.

Parameters

<code>__o</code>	Operator.
<code>__i</code>	iterator referencing object.

Definition at line 123 of file `for_each_selectors.h`.

#### 4.101.3 Member Data Documentation

4.101.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator [inherited]`

\_\_iterator on last element processed; needed for some algorithms (e. g. `std::transform()`).

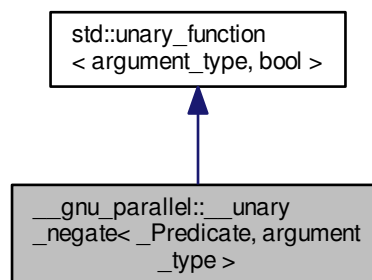
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 4.102 `__gnu_parallel::__unary_negate<_Predicate, argument_type>` Class Template Reference

Inheritance diagram for `__gnu_parallel::__unary_negate<_Predicate, argument_type>`:



### Public Types

- typedef [argument\\_type](#) **argument\_type**
- typedef bool [result\\_type](#)

### Public Member Functions

- **\_\_unary\_negate** (const [\\_Predicate](#) &\_\_x)
- bool **operator()** (const [argument\\_type](#) &\_\_x)

### Protected Attributes

- [\\_Predicate](#) **\_M\_pred**

#### 4.102.1 Detailed Description

```
template<typename _Predicate, typename argument_type> class __gnu_parallel::__unary_negate<_Predicate, argument_type>
```

Similar to `std::unary_negate`, but giving the argument types explicitly.

Definition at line 173 of file `parallel/base.h`.

#### 4.102.2 Member Typedef Documentation

4.102.2.1 typedef [argument\\_type](#) `std::unary_function< argument\_type, bool>::argument_type` [inherited]

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

4.102.2.2 typedef bool `std::unary_function< argument\_type, bool>::result_type` [inherited]

`result_type` is the return type

Definition at line 107 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

#### 4.103 `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>` Struct Template Reference

##### Public Types

- typedef `_TraitsType::difference_type` **DifferenceType**
- typedef `std::iterator_traits<_RAIter>` **\_TraitsType**
- typedef `_TraitsType::value_type` **ValueType**

##### Public Member Functions

- [\\_DRandomShufflingGlobalData](#) (`_RAIter` &\_\_source)

## Public Attributes

- `_ThreadIndex * _M_bin_proc`
- `_DifferenceType ** _M_dist`
- `int _M_num_bins`
- `int _M_num_bits`
- `_RAIter & _M_source`
- `_DifferenceType * _M_starts`
- `_ValueType ** _M_temporaries`

## 4.103.1 Detailed Description

`template<typename _RAIter> struct __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>`

Data known to every thread participating in `__gnu_parallel::__parallel_random_shuffle()`.

Definition at line 52 of file `random_shuffle.h`.

## 4.103.2 Constructor &amp; Destructor Documentation

4.103.2.1 `template<typename _RAIter> __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::__DRandomShufflingGlobalData( _RAIter & __source ) [inline]`

Constructor.

Definition at line 83 of file `random_shuffle.h`.

## 4.103.3 Member Data Documentation

4.103.3.1 `template<typename _RAIter> _ThreadIndex* __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::__M_bin_proc`

Number of the thread that will further process the corresponding bin.

Definition at line 74 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`.

4.103.3.2 `template<typename _RAIter> _DifferenceType** __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::__M_dist`

Two-dimensional array to hold the thread-bin distribution.

Dimensions `( _M_num_threads + 1 ) __x ( _M_num_bins + 1 )`.

Definition at line 67 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs←_pu()`.

4.103.3.3 `template<typename _RAIter> int __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::__M_num_bins`

Number of bins to distribute to.

Definition at line 77 of file `random_shuffle.h`.



Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs↵  
_pu()`.

**4.103.3.4** `template<typename _RAIter> int __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bits`

Number of bits needed to address the bins.

Definition at line 80 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs↵  
_pu()`.

**4.103.3.5** `template<typename _RAIter> _RAIter& __gnu_parallel::_DRandomShufflingGlobalData< _RAIter  
>::_M_source`

Begin iterator of the `__source`.

Definition at line 59 of file `random_shuffle.h`.

**4.103.3.6** `template<typename _RAIter> _DifferenceType* __gnu_parallel::_DRandomShufflingGlobalData< _RAIter  
>::_M_starts`

Start indexes of the threads' `__chunks`.

Definition at line 70 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs↵  
_pu()`.

**4.103.3.7** `template<typename _RAIter> _ValueType** __gnu_parallel::_DRandomShufflingGlobalData< _RAIter  
>::_M_temporaries`

Temporary arrays for each thread.

Definition at line 62 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`.

The documentation for this struct was generated from the following file:

- [random\\_shuffle.h](#)

## 4.104 `__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >` Struct Template Reference

### Public Attributes

- [\\_BinIndex \\_\\_bins\\_end](#)
- [\\_BinIndex \\_M\\_bins\\_begin](#)
- [int \\_M\\_num\\_threads](#)
- [\\_DRandomShufflingGlobalData< \\_RAIter > \\* \\_M\\_sd](#)
- [uint32\\_t \\_M\\_seed](#)

### 4.104.1 Detailed Description

`template<typename _RAIter, typename _RandomNumberGenerator> struct __gnu_parallel::_DRSSorterPU< _RAIter, _Random↵  
NumberGenerator >`

Local data for a thread participating in `__gnu_parallel::__parallel_random_shuffle()`.

Definition at line 91 of file `random_shuffle.h`.

#### 4.104.2 Member Data Documentation

4.104.2.1 `template<typename _RAIter, typename _RandomNumberGenerator> _BinIndex __gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::_bins_end`

End index for bins taken care of by this thread.

Definition at line 100 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`.

4.104.2.2 `template<typename _RAIter, typename _RandomNumberGenerator> _BinIndex __gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::_M_bins_begin`

Begin index for bins taken care of by this thread.

Definition at line 97 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`.

4.104.2.3 `template<typename _RAIter, typename _RandomNumberGenerator> int __gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::_M_num_threads`

Number of threads participating in total.

Definition at line 94 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs←_pu()`.

4.104.2.4 `template<typename _RAIter, typename _RandomNumberGenerator> _DRandomShufflingGlobalData<_RAIter>* __gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::_M_sd`

Pointer to global data.

Definition at line 106 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs←_pu()`.

4.104.2.5 `template<typename _RAIter, typename _RandomNumberGenerator> uint32_t __gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::_M_seed`

Random `_M_seed` for this thread.

Definition at line 103 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs←_pu()`.

The documentation for this struct was generated from the following file:

- [random\\_shuffle.h](#)

#### 4.105 `__gnu_parallel::_DummyReduct` Struct Reference

### Public Member Functions

- bool **operator()** (bool, bool) const

#### 4.105.1 Detailed Description

Reduction function doing nothing.

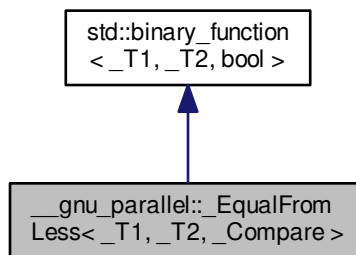
Definition at line 298 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

### 4.106 `__gnu_parallel::_EqualFromLess<_T1, _T2, _Compare>` Class Template Reference

Inheritance diagram for `__gnu_parallel::_EqualFromLess<_T1, _T2, _Compare>`:



### Public Types

- typedef `_T1` [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef `_T2` [second\\_argument\\_type](#)

### Public Member Functions

- **\_EqualFromLess** (`_Compare` &`__comp`)
- bool **operator()** (const `_T1` &`__a`, const `_T2` &`__b`)

#### 4.106.1 Detailed Description

```
template<typename _T1, typename _T2, typename _Compare>class __gnu_parallel::_EqualFromLess<_T1, _T2, _Compare>
```

Constructs predicate for equality from strict weak ordering predicate.

Definition at line 157 of file `parallel/base.h`.

## 4.106.2 Member Typedef Documentation

4.106.2.1 `typedef _T1 std::binary_function<_T1, _T2, bool>::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.106.2.2 `typedef bool std::binary_function<_T1, _T2, bool>::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.106.2.3 `typedef _T2 std::binary_function<_T1, _T2, bool>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

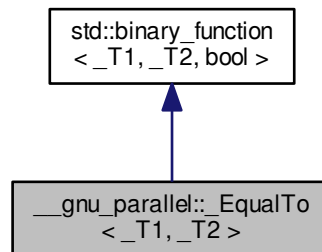
Definition at line 120 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

4.107 `__gnu_parallel::_EqualTo<_T1, _T2>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::_EqualTo<_T1, _T2>`:



## Public Types

- `typedef _T1` [first\\_argument\\_type](#)
- `typedef bool` [result\\_type](#)
- `typedef _T2` [second\\_argument\\_type](#)

## Public Member Functions

- `bool` **operator()** (`const _T1 &__t1`, `const _T2 &__t2`) `const`

#### 4.107.1 Detailed Description

`template<typename _T1, typename _T2> struct __gnu_parallel::_EqualTo< _T1, _T2 >`

Similar to `std::equal_to`, but allows two different types.

Definition at line 244 of file `parallel/base.h`.

#### 4.107.2 Member Typedef Documentation

4.107.2.1 `typedef _T1 std::binary_function< _T1, _T2, bool >::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.107.2.2 `typedef bool std::binary_function< _T1, _T2, bool >::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.107.2.3 `typedef _T2 std::binary_function< _T1, _T2, bool >::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

### 4.108 `__gnu_parallel::_GuardedIterator< _RAIter, _Compare >` Class Template Reference

#### Public Member Functions

- `_GuardedIterator` (`_RAIter __begin`, `_RAIter __end`, `_Compare &__comp`)
- `operator _RAIter` ()
- `std::iterator_traits< _RAIter >::value_type & operator*` ()
- `_GuardedIterator< _RAIter, _Compare > & operator++` ()

#### Friends

- `bool operator<` (`_GuardedIterator< _RAIter, _Compare > &__bi1`, `_GuardedIterator< _RAIter, _Compare > &__bi2`)
- `bool operator<=` (`_GuardedIterator< _RAIter, _Compare > &__bi1`, `_GuardedIterator< _RAIter, _Compare > &__bi2`)

#### 4.108.1 Detailed Description

`template<typename _RAIter, typename _Compare> class __gnu_parallel::_GuardedIterator< _RAIter, _Compare >`

`_Iterator` wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons.

The implicit supremum comes with a performance cost.

Deriving from `_RAIter` is not possible since `_RAIter` need not be a class.

Definition at line 73 of file `multiway_merge.h`.

#### 4.108.2 Constructor & Destructor Documentation

4.108.2.1 `template<typename _RAIter, typename _Compare> __gnu_parallel::__GuardedIterator<_RAIter, _Compare>::__GuardedIterator( _RAIter __begin, _RAIter __end, _Compare & __comp ) [inline]`

Constructor. Sets iterator to beginning of sequence.

Parameters

<code>__begin</code>	Begin iterator of sequence.
<code>__end</code>	End iterator of sequence.
<code>__comp</code>	Comparator provided for associated overloaded compare operators.

Definition at line 91 of file `multiway_merge.h`.

#### 4.108.3 Member Function Documentation

4.108.3.1 `template<typename _RAIter, typename _Compare> __gnu_parallel::__GuardedIterator<_RAIter, _Compare>::operator _RAIter( ) [inline]`

Convert to wrapped iterator.

Returns

Wrapped iterator.

Definition at line 112 of file `multiway_merge.h`.

4.108.3.2 `template<typename _RAIter, typename _Compare> std::iterator_traits<_RAIter>::value_type& __gnu_parallel::__GuardedIterator<_RAIter, _Compare>::operator*( ) [inline]`

Dereference operator.

Returns

Referenced element.

Definition at line 107 of file `multiway_merge.h`.

4.108.3.3 `template<typename _RAIter, typename _Compare> __GuardedIterator<_RAIter, _Compare>& __gnu_parallel::__GuardedIterator<_RAIter, _Compare>::operator++( ) [inline]`

Pre-increment operator.

Returns

This.

Definition at line 98 of file `multiway_merge.h`.

#### 4.108.4 Friends And Related Function Documentation

4.108.4.1 `template<typename _RAIter, typename _Compare > bool operator< ( _GuardedIterator< _RAIter, _Compare > & __bi1, _GuardedIterator< _RAIter, _Compare > & __bi2 ) [friend]`

Compare two elements referenced by guarded iterators.

## 4.109 `__gnu_parallel::_IteratorPair<_Iterator1, _Iterator2, _IteratorCategory>` > Class Template Reference 627

### Parameters

<code>__bi1</code>	First iterator.
<code>__bi2</code>	Second iterator.

### Returns

`true` if less.

Definition at line 120 of file `multiway_merge.h`.

4.108.4.2 `template<typename _RAIter, typename _Compare> bool operator<= ( _GuardedIterator<_RAIter, _Compare> & __bi1, _GuardedIterator<_RAIter, _Compare> & __bi2 ) [friend]`

Compare two elements referenced by guarded iterators.

### Parameters

<code>__bi1</code>	First iterator.
<code>__bi2</code>	Second iterator.

### Returns

`True` if less equal.

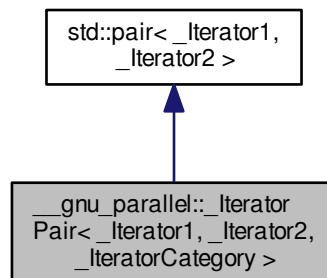
Definition at line 135 of file `multiway_merge.h`.

The documentation for this class was generated from the following file:

- [multiway\\_merge.h](#)

## 4.109 `__gnu_parallel::_IteratorPair<_Iterator1, _Iterator2, _IteratorCategory>` > Class Template Reference

Inheritance diagram for `__gnu_parallel::_IteratorPair<_Iterator1, _Iterator2, _IteratorCategory>`:



### Public Types

- `typedef std::iterator_traits<_Iterator1> _TraitsType`



- typedef `_TraitsType::difference_type` **difference\_type**
- typedef `_Iterator1` **first\_type**
- typedef `_IteratorCategory` **iterator\_category**
- typedef `_IteratorPair` \* **pointer**
- typedef `_IteratorPair` & **reference**
- typedef `_Iterator2` **second\_type**
- typedef void **value\_type**

#### Public Member Functions

- `_IteratorPair` (const `_Iterator1` &\_\_first, const `_Iterator2` &\_\_second)
- **operator** `_Iterator2` () const
- `_IteratorPair` **operator+** (difference\_type \_\_delta) const
- `_IteratorPair` & **operator++** ()
- const `_IteratorPair` **operator++** (int)
- difference\_type **operator-** (const `_IteratorPair` &\_\_other) const
- `_IteratorPair` & **operator--** ()
- const `_IteratorPair` **operator--** (int)
- `_IteratorPair` & **operator=** (const `_IteratorPair` &\_\_other)
- void **swap** (`pair` &\_\_p) noexcept(noexcept(swap(`first`, \_\_p.first))&&noexcept(swap(`second`, \_\_p.second)))

#### Public Attributes

- `_Iterator1` `first`
- `_Iterator2` `second`

#### 4.109.1 Detailed Description

```
template<typename _Iterator1, typename _Iterator2, typename _IteratorCategory>class __gnu_parallel::_IteratorPair< _Iterator1, ↵
_iterator2, _IteratorCategory >
```

A pair of iterators. The usual iterator operations are applied to both child iterators.

Definition at line 45 of file `iterator.h`.

#### 4.109.2 Member Typedef Documentation

4.109.2.1 `typedef _Iterator2 std::pair< _Iterator1 , _Iterator2 >::second_type` [inherited]

`first_type` is the first bound type

Definition at line 99 of file `stl_pair.h`.

#### 4.109.3 Member Data Documentation

4.109.3.1 `_Iterator1 std::pair< _Iterator1 , _Iterator2 >::first` [inherited]

`second_type` is the second bound type

Definition at line 101 of file `stl_pair.h`.

4.109.3.2 `_Iterator2 std::pair<_Iterator1, _Iterator2>::second` [inherited]

`first` is a copy of the first object

Definition at line 102 of file `stl_pair.h`.

The documentation for this class was generated from the following file:

- [iterator.h](#)

## 4.110 `__gnu_parallel::_IteratorTriple<_Iterator1, _Iterator2, _Iterator3, _IteratorCategory>` Class Template Reference

### Public Types

- typedef `std::iterator_traits<_Iterator1>::difference_type` **difference\_type**
- typedef `_IteratorCategory` **iterator\_category**
- typedef `_IteratorTriple` \* **pointer**
- typedef `_IteratorTriple` & **reference**
- typedef void **value\_type**

### Public Member Functions

- **\_IteratorTriple** (const `_Iterator1` &\_\_first, const `_Iterator2` &\_\_second, const `_Iterator3` &\_\_third)
- **operator \_Iterator3** () const
- `_IteratorTriple` **operator+** (difference\_type \_\_delta) const
- `_IteratorTriple` & **operator++** ()
- const `_IteratorTriple` **operator++** (int)
- difference\_type **operator-** (const `_IteratorTriple` &\_\_other) const
- `_IteratorTriple` & **operator--** ()
- const `_IteratorTriple` **operator--** (int)
- `_IteratorTriple` & **operator=** (const `_IteratorTriple` &\_\_other)

### Public Attributes

- `_Iterator1` **\_M\_first**
- `_Iterator2` **\_M\_second**
- `_Iterator3` **\_M\_third**

### 4.110.1 Detailed Description

```
template<typename _Iterator1, typename _Iterator2, typename _Iterator3, typename _IteratorCategory>class __gnu_parallel::_IteratorTriple<_Iterator1, _Iterator2, _Iterator3, _IteratorCategory>
```

A triple of iterators. The usual iterator operations are applied to all three child iterators.

Definition at line 120 of file `iterator.h`.

The documentation for this class was generated from the following file:

- [iterator.h](#)

#### 4.111 `__gnu_parallel::__Job<_DifferenceTp>` Struct Template Reference

##### Public Types

- `typedef _DifferenceTp _DifferenceType`

##### Public Attributes

- `volatile _DifferenceType \_M\_first`
- `volatile _DifferenceType \_M\_last`
- `volatile _DifferenceType \_M\_load`

##### 4.111.1 Detailed Description

`template<typename _DifferenceTp>struct __gnu_parallel::__Job<_DifferenceTp>`

One `__job` for a certain thread.

Definition at line 54 of file `workstealing.h`.

##### 4.111.2 Member Data Documentation

4.111.2.1 `template<typename _DifferenceTp> volatile _DifferenceType __gnu_parallel::__Job<_DifferenceTp>::_M_first`

First element.

Changed by owning and stealing thread. By stealing thread, always incremented.

Definition at line 62 of file `workstealing.h`.

Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`.

4.111.2.2 `template<typename _DifferenceTp> volatile _DifferenceType __gnu_parallel::__Job<_DifferenceTp>::_M_last`

Last element.

Changed by owning thread only.

Definition at line 67 of file `workstealing.h`.

Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`.

4.111.2.3 `template<typename _DifferenceTp> volatile _DifferenceType __gnu_parallel::__Job<_DifferenceTp>::_M_load`

Number of elements, i.e. `_M_last - _M_first + 1`.

Changed by owning thread only.

Definition at line 72 of file `workstealing.h`.

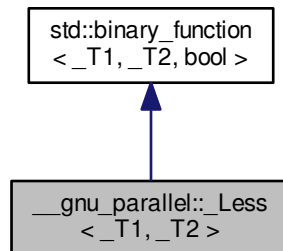
Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`.

The documentation for this struct was generated from the following file:

- [workstealing.h](#)

4.112 `__gnu_parallel::_Less<_T1,_T2>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::_Less<_T1,_T2>`:



## Public Types

- typedef `_T1` `first_argument_type`
- typedef `bool` `result_type`
- typedef `_T2` `second_argument_type`

## Public Member Functions

- `bool operator() (const _T1 &__t1, const _T2 &__t2) const`
- `bool operator() (const _T2 &__t2, const _T1 &__t1) const`

## 4.112.1 Detailed Description

```
template<typename _T1, typename _T2> struct __gnu_parallel::_Less<_T1, _T2 >
```

Similar to `std::less`, but allows two different types.

Definition at line 252 of file `parallel/base.h`.

## 4.112.2 Member Typedef Documentation

4.112.2.1 typedef `_T1` `std::binary_function<_T1, _T2, bool>::first_argument_type` `[inherited]`

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.112.2.2 typedef `bool` `std::binary_function<_T1, _T2, bool>::result_type` `[inherited]`

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

#### 4.112.2.3 `typedef _T2 std::binary_function< _T1, _T2, bool >::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

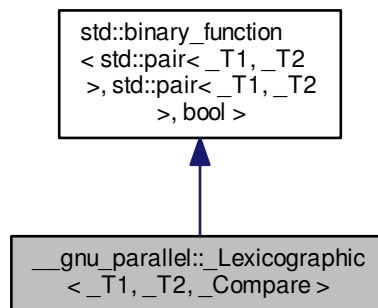
Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

### 4.113 `__gnu_parallel::_Lexicographic< _T1, _T2, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_Lexicographic< _T1, _T2, _Compare >`:



#### Public Types

- typedef `std::pair< _T1, _T2 >` `first_argument_type`
- typedef `bool` `result_type`
- typedef `std::pair< _T1, _T2 >` `second_argument_type`

#### Public Member Functions

- `_Lexicographic` (`_Compare` &`__comp`)
- `bool operator()` (`const std::pair< _T1, _T2 >` &`__p1`, `const std::pair< _T1, _T2 >` &`__p2`) `const`

#### 4.113.1 Detailed Description

`template<typename _T1, typename _T2, typename _Compare> class __gnu_parallel::_Lexicographic< _T1, _T2, _Compare >`

Compare \_\_a pair of types lexicographically, ascending.

Definition at line 53 of file `multiseq_selection.h`.

## 4.113.2 Member Typedef Documentation

4.113.2.1 `typedef std::pair<_T1, _T2> std::binary_function< std::pair<_T1, _T2>, std::pair<_T1, _T2>, bool >::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.113.2.2 `typedef bool std::binary_function< std::pair<_T1, _T2>, std::pair<_T1, _T2>, bool >::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.113.2.3 `typedef std::pair<_T1, _T2> std::binary_function< std::pair<_T1, _T2>, std::pair<_T1, _T2>, bool >::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

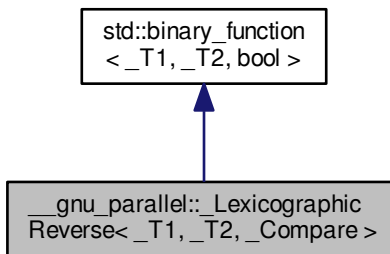
Definition at line 120 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [multiseq\\_selection.h](#)

4.114 `__gnu_parallel::_LexicographicReverse<_T1, _T2, _Compare>` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LexicographicReverse<_T1, _T2, _Compare>`:



## Public Types

- `typedef _T1` [first\\_argument\\_type](#)
- `typedef bool` [result\\_type](#)
- `typedef _T2` [second\\_argument\\_type](#)

## Public Member Functions

- **\_LexicographicReverse** (\_Compare &\_\_comp)
- bool **operator()** (const [std::pair](#)<\_T1, \_T2> &\_\_p1, const [std::pair](#)<\_T1, \_T2> &\_\_p2) const

## 4.114.1 Detailed Description

```
template<typename _T1, typename _T2, typename _Compare>class __gnu_parallel::_LexicographicReverse<_T1, _T2, _Compare >
```

Compare \_\_a pair of types lexicographically, descending.

Definition at line 80 of file [multiseq\\_selection.h](#).

## 4.114.2 Member Typedef Documentation

4.114.2.1 `typedef _T1 std::binary_function<_T1, _T2, bool>::first_argument_type` `[inherited]`

`first_argument_type` is the type of the first argument

Definition at line 117 of file [stl\\_function.h](#).

4.114.2.2 `typedef bool std::binary_function<_T1, _T2, bool>::result_type` `[inherited]`

`result_type` is the return type

Definition at line 123 of file [stl\\_function.h](#).

4.114.2.3 `typedef _T2 std::binary_function<_T1, _T2, bool>::second_argument_type` `[inherited]`

`second_argument_type` is the type of the second argument

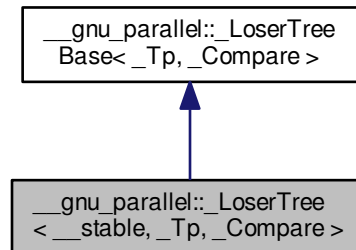
Definition at line 120 of file [stl\\_function.h](#).

The documentation for this class was generated from the following file:

- [multiseq\\_selection.h](#)

4.115 `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >`:



## Public Member Functions

- **`_LoserTree`** (unsigned int `__k`, `_Compare` `__comp`)
- void `__delete_min_insert` (`_Tp` `__key`, bool `__sup`)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int `__root`)
- void `__insert_start` (const `_Tp` & `__key`, int `__source`, bool `__sup`)

## Protected Attributes

- `_Compare` `_M_comp`
- bool `_M_first_insert`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- unsigned int `_M_log_k`
- `_Loser` \* `_M_losers`
- unsigned int `_M_offset`

## 4.115.1 Detailed Description

```
template<bool __stable, typename _Tp, typename _Compare>class __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >
```

Stable `_LoserTree` variant.

Provides the stable implementations of `insert_start`, `__init_winner`, `__init` and `__delete_min_insert`.

Unstable variant is done using partial specialisation below.

Definition at line 169 of file `losertree.h`.



#### 4.115.2 Member Function Documentation

4.115.2.1 `template<bool __stable, typename _Tp, typename _Compare > void __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >::_delete_min_insert ( _Tp __key, bool __sup ) [inline]`

Delete the smallest element and insert a new element from the previously smallest element's sequence.

This implementation is stable.

Definition at line 222 of file losertree.h.

References `std::swap()`.

4.115.2.2 `template<typename _Tp, typename _Compare > int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_get_min_source ( ) [inline],[inherited]`

##### Returns

the index of the sequence with the smallest element.

Definition at line 155 of file losertree.h.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`.

4.115.2.3 `template<typename _Tp, typename _Compare > void __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start ( const _Tp & __key, int __source, bool __sup ) [inline],[inherited]`

Initializes the sequence "`_M_source`" with the element "`__key`".

##### Parameters

<code>__key</code>	the element to insert
<code>__source</code>	<code>__index</code> of the <code>__source</code> <code>__sequence</code>
<code>__sup</code>	flag that determines whether the value to insert is an explicit <code>__supremum</code> .

Definition at line 134 of file losertree.h.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup`.

#### 4.115.3 Member Data Documentation

4.115.3.1 `template<typename _Tp, typename _Compare > _Compare __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_comp [protected],[inherited]`

`_Compare` to use.

Definition at line 78 of file losertree.h.

4.115.3.2 `template<typename _Tp, typename _Compare > bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert [protected],[inherited]`

State flag that determines whether the `_LoserTree` is empty.

Only used for building the `_LoserTree`.

Definition at line 85 of file losertree.h.

4.115.3.3 `template<typename _Tp, typename _Compare> unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k` [protected], [inherited]

`log_2{ _M_k }`

Definition at line 72 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

4.115.3.4 `template<typename _Tp, typename _Compare> _Loser* __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers` [protected], [inherited]

`_LoserTree` `__elements`.

Definition at line 75 of file `losertree.h`.

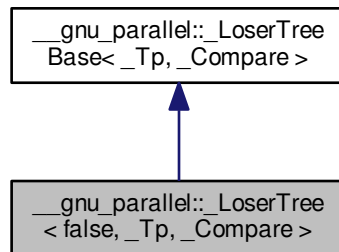
Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~~LoserTreeBase()`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 4.116 `__gnu_parallel::_LoserTree< false, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`:



### Public Member Functions

- **`_LoserTree`** (`unsigned int __k, _Compare __comp`)
- `void __delete_min_insert` (`_Tp __key, bool __sup`)
- `int __get_min_source` ()
- `void __init` ()
- `unsigned int __init_winner` (`unsigned int __root`)
- `void __insert_start` (`const _Tp &__key, int __source, bool __sup`)

### Protected Attributes

- `_Compare` `_M_comp`

- [bool \\_M\\_first\\_insert](#)
- [unsigned int \\_M\\_ik](#)
- [unsigned int \\_M\\_k](#)
- [unsigned int \\_M\\_log\\_k](#)
- [\\_Loser \\* \\_M\\_losers](#)
- [unsigned int \\_M\\_offset](#)

#### 4.116.1 Detailed Description

`template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTree< false, _Tp, _Compare >`

Unstable `_LoserTree` variant.

Stability (non-stable here) is selected with partial specialization.

Definition at line 261 of file `losertree.h`.

#### 4.116.2 Member Function Documentation

4.116.2.1 `template<typename _Tp, typename _Compare > void __gnu_parallel::_LoserTree< false, _Tp, _Compare >::_delete_min_insert( _Tp __key, bool __sup ) [inline]`

Delete the `_M_key` smallest element and insert the element `__key` instead.

Parameters

<code>__key</code>	the <code>_M_key</code> to insert
<code>__sup</code>	true iff <code>__key</code> is an explicitly marked supremum

Definition at line 324 of file `losertree.h`.

References `std::swap()`.

4.116.2.2 `template<typename _Tp, typename _Compare > int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_get_min_source( ) [inline],[inherited]`

Returns

the index of the sequence with the smallest element.

Definition at line 155 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`.

4.116.2.3 `template<typename _Tp, typename _Compare > unsigned int __gnu_parallel::_LoserTree< false, _Tp, _Compare >::_init_winner( unsigned int __root ) [inline]`

Computes the winner of the competition at position "`__root`".

Called recursively (starting at 0) to build the initial tree.

Parameters

<code>__root</code>	<code>__index</code> of the "game" to start.
---------------------	--

Definition at line 284 of file `losertree.h`.

4.116.2.4 `template<typename _Tp, typename _Compare > void __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start( const _Tp & __key, int __source, bool __sup )` `[inline]`, `[inherited]`

Initializes the sequence "`_M_source`" with the element "`__key`".

## Parameters

<code>__key</code>	the element to insert
<code>__source</code>	<code>__index</code> of the <code>__source</code> <code>__sequence</code>
<code>__sup</code>	flag that determines whether the value to insert is an explicit <code>__supremum</code> .

Definition at line 134 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup`.

## 4.116.3 Member Data Documentation

4.116.3.1 `template<typename _Tp, typename _Compare > _Compare __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_comp` `[protected]`, `[inherited]`

`_Compare` to use.

Definition at line 78 of file `losertree.h`.

4.116.3.2 `template<typename _Tp, typename _Compare > bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert` `[protected]`, `[inherited]`

State flag that determines whether the `_LoserTree` is empty.

Only used for building the `_LoserTree`.

Definition at line 85 of file `losertree.h`.

4.116.3.3 `template<typename _Tp, typename _Compare > unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k` `[protected]`, `[inherited]`

`log_2{_M_k}`

Definition at line 72 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

4.116.3.4 `template<typename _Tp, typename _Compare > _Loser* __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers` `[protected]`, `[inherited]`

`_LoserTree` `__elements`.

Definition at line 75 of file `losertree.h`.

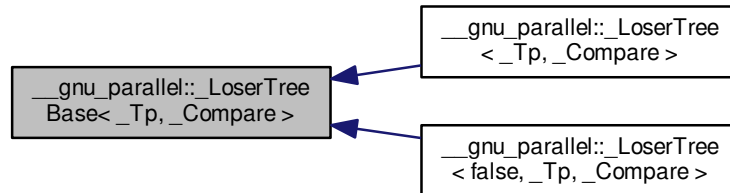
Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~~LoserTreeBase()`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.117 `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>`:



## Classes

- struct [\\_Loser](#)

## Public Member Functions

- [\\_LoserTreeBase](#) (unsigned int \_\_k, \_Compare \_\_comp)
- [~\\_LoserTreeBase](#) ()
- int [\\_\\_get\\_min\\_source](#) ()
- void [\\_\\_insert\\_start](#) (const \_Tp &\_\_key, int \_\_source, bool \_\_sup)

## Protected Attributes

- [\\_Compare](#) [\\_M\\_comp](#)
- bool [\\_M\\_first\\_insert](#)
- unsigned int [\\_M\\_ik](#)
- unsigned int [\\_M\\_k](#)
- unsigned int [\\_M\\_log\\_k](#)
- [\\_Loser](#) \* [\\_M\\_losers](#)
- unsigned int [\\_M\\_offset](#)

## 4.117.1 Detailed Description

```
template<typename _Tp, typename _Compare>class __gnu_parallel::_LoserTreeBase<_Tp, _Compare>
```

Guarded loser/tournament tree.

The smallest element is at the top.

Guarding is done explicitly through one flag `_M_sup` per element, `inf` is not needed due to a better initialization routine. This is a well-performing variant.

## Parameters

<code>_Tp</code>	the element type
<code>_Compare</code>	the comparator to use, defaults to <code>std::less&lt;_Tp&gt;</code>

Definition at line 55 of file `losertree.h`.

## 4.117.2 Constructor &amp; Destructor Documentation

4.117.2.1 `template<typename _Tp, typename _Compare> __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::__LoserTreeBase ( unsigned int __k, _Compare __comp ) [inline]`

The constructor.

## Parameters

<code>__k</code>	The number of sequences to merge.
<code>__comp</code>	The comparator to use.

Definition at line 94 of file `losertree.h`.

References `__gnu_parallel::__rd_log2()`, and `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::__M_log_k`.

4.117.2.2 `template<typename _Tp, typename _Compare> __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::~~LoserTreeBase ( ) [inline]`

The destructor.

Definition at line 118 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::__M_losers`.

## 4.117.3 Member Function Documentation

4.117.3.1 `template<typename _Tp, typename _Compare> int __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::__get_min_source ( ) [inline]`

## Returns

the index of the sequence with the smallest element.

Definition at line 155 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::__Loser::__M_source`.

4.117.3.2 `template<typename _Tp, typename _Compare> void __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::__insert_start ( const _Tp & __key, int __source, bool __sup ) [inline]`

Initializes the sequence "`_M_source`" with the element "`__key`".

## Parameters

<code>__key</code>	the element to insert
<code>__source</code>	<code>__index</code> of the <code>__source</code> <code>__sequence</code>

<code>__sup</code>	flag that determines whether the value to insert is an explicit <code>__supremum</code> .
--------------------	---

Definition at line 134 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup`.

#### 4.117.4 Member Data Documentation

4.117.4.1 `template<typename _Tp, typename _Compare > _Compare __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_comp` `[protected]`

`_Compare` to use.

Definition at line 78 of file `losertree.h`.

4.117.4.2 `template<typename _Tp, typename _Compare > bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert` `[protected]`

State flag that determines whether the `_LoserTree` is empty.

Only used for building the `_LoserTree`.

Definition at line 85 of file `losertree.h`.

4.117.4.3 `template<typename _Tp, typename _Compare > unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k` `[protected]`

`log_2{_M_k}`

Definition at line 72 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

4.117.4.4 `template<typename _Tp, typename _Compare > _Loser* __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers` `[protected]`

`_LoserTree` `__elements`.

Definition at line 75 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~~_LoserTreeBase()`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 4.118 `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser` Struct Reference

### Public Attributes

- `_Tp` `_M_key`
- `int` `_M_source`
- `bool` `_M_sup`

#### 4.118.1 Detailed Description



```
template<typename _Tp, typename _Compare>struct __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser
```

Internal representation of a `_LoserTree` element.

Definition at line 59 of file `losertree.h`.

#### 4.118.2 Member Data Documentation

4.118.2.1 `template<typename _Tp, typename _Compare > _Tp __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key`

`_M_key` of the element in the `_LoserTree`.

Definition at line 66 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`.

4.118.2.2 `template<typename _Tp, typename _Compare > int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`

`__index` of the `__source` `__sequence`.

Definition at line 64 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_get_min_source()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`.

4.118.2.3 `template<typename _Tp, typename _Compare > bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup`

flag, true iff this is a "maximum" `__sentinel`.

Definition at line 62 of file `losertree.h`.

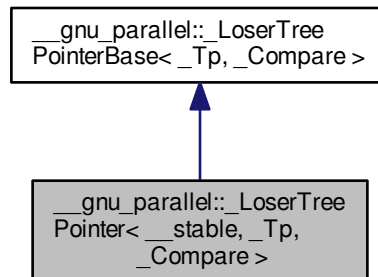
Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`.

The documentation for this struct was generated from the following file:

- [losertree.h](#)

4.119 `__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >`:



## Public Member Functions

- `_LoserTreePointer` (unsigned int \_\_k, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- void `__delete_min_insert` (const \_Tp &\_\_key, bool \_\_sup)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int \_\_root)
- void `__insert_start` (const \_Tp &\_\_key, int \_\_source, bool \_\_sup)

## Protected Attributes

- \_Compare `_M_comp`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- [\\_Loser](#) \* `_M_losers`
- unsigned int `_M_offset`

## 4.119.1 Detailed Description

```
template<bool __stable, typename _Tp, typename _Compare>class __gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >
```

Stable `_LoserTree` implementation.

The unstable variant is implemented using partial instantiation below.

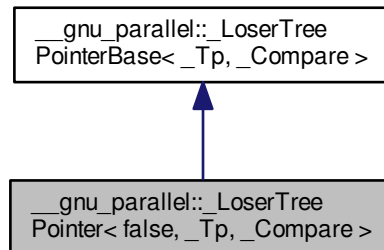
Definition at line 409 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

#### 4.120 `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >`:



##### Public Member Functions

- `_LoserTreePointer` (unsigned int \_\_k, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- void `__delete_min_insert` (const \_Tp &\_\_key, bool \_\_sup)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int \_\_root)
- void `__insert_start` (const \_Tp &\_\_key, int \_\_source, bool \_\_sup)

##### Protected Attributes

- `_Compare` `_M_comp`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- [\\_Loser](#) \* `_M_losers`
- unsigned int `_M_offset`

##### 4.120.1 Detailed Description

`template<typename _Tp, typename _Compare>class __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >`

Unstable `_LoserTree` implementation.

The stable variant is above.

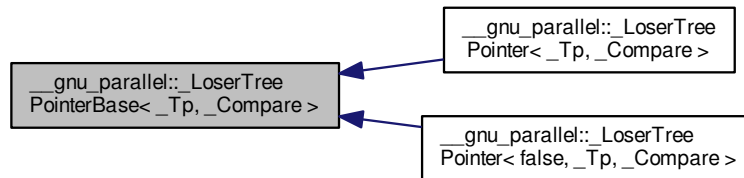
Definition at line 491 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.121 `__gnu_parallel::_LoserTreePointerBase<_Tp, _Compare>` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointerBase<_Tp, _Compare>`:



## Classes

- struct [\\_Loser](#)

## Public Member Functions

- **`_LoserTreePointerBase`** (unsigned int \_\_k, \_Compare \_\_comp=[std::less](#)<\_Tp>())
- int **`__get_min_source`** ()
- void **`__insert_start`** (const \_Tp &\_\_key, int \_\_source, bool \_\_sup)

## Protected Attributes

- \_Compare **`_M_comp`**
- unsigned int **`_M_ik`**
- unsigned int **`_M_k`**
- [\\_Loser](#) \* **`_M_losers`**
- unsigned int **`_M_offset`**

## 4.121.1 Detailed Description

```
template<typename _Tp, typename _Compare>class __gnu_parallel::_LoserTreePointerBase<_Tp, _Compare>
```

Base class of `_Loser` Tree implementation using pointers.

Definition at line 357 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.122 `__gnu_parallel::_LoserTreePointerBase<_Tp, _Compare>::_Loser` Struct Reference

## Public Attributes

- const \_Tp \* **`_M_keyp`**

- int **\_M\_source**
- bool **\_M\_sup**

#### 4.122.1 Detailed Description

template<typename \_Tp, typename \_Compare>struct \_\_gnu\_parallel::\_LoserTreePointerBase<\_Tp, \_Compare >::\_Loser

Internal representation of \_LoserTree \_\_elements.

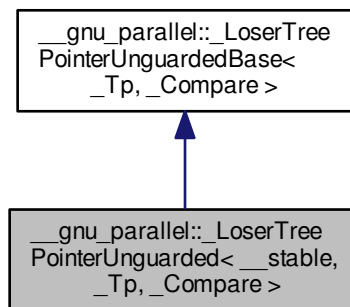
Definition at line 361 of file losertree.h.

The documentation for this struct was generated from the following file:

- [losertree.h](#)

#### 4.123 \_\_gnu\_parallel::\_LoserTreePointerUnguarded<\_\_stable, \_Tp, \_Compare > Class Template Reference

Inheritance diagram for \_\_gnu\_parallel::\_LoserTreePointerUnguarded<\_\_stable, \_Tp, \_Compare >:



#### Public Member Functions

- **\_LoserTreePointerUnguarded** (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=[std::less](#)<\_Tp>())
- void **\_\_delete\_min\_insert** (const \_Tp &\_\_key, bool \_\_sup)
- int **\_\_get\_min\_source** ()
- void **\_\_init** ()
- unsigned int **\_\_init\_winner** (unsigned int \_\_root)
- void **\_\_insert\_start** (const \_Tp &\_\_key, int \_\_source, bool)

#### Protected Attributes

- \_Compare **\_M\_comp**
- unsigned int **\_M\_ik**

- unsigned int `_M_k`
- `_Loser * _M_losers`
- unsigned int `_M_offset`

## 4.123.1 Detailed Description

```
template<bool __stable, typename _Tp, typename _Compare>class __gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp,
__Compare >
```

Stable unguarded `_LoserTree` variant storing pointers.

Unstable variant is implemented below using partial specialization.

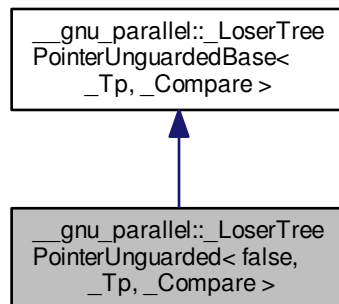
Definition at line 891 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.124 `__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >`:



## Public Member Functions

- **`_LoserTreePointerUnguarded`** (unsigned int `__k`, const `_Tp` &`__sentinel`, `_Compare` `__comp=std::less< _Tp >()`)
- void **`__delete_min_insert`** (const `_Tp` &`__key`, bool `__sup`)
- int **`__get_min_source`** ()
- void **`__init`** ()
- unsigned int **`__init_winner`** (unsigned int `__root`)
- void **`__insert_start`** (const `_Tp` &`__key`, int `__source`, bool)

## Protected Attributes

- `_Compare` **`_M_comp`**
- unsigned int **`_M_ik`**
- unsigned int **`_M_k`**
- `_Loser` \* **`_M_losers`**
- unsigned int **`_M_offset`**

## 4.124.1 Detailed Description

```
template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >
```

Unstable unguarded `_LoserTree` variant storing pointers.

Stable variant is above.

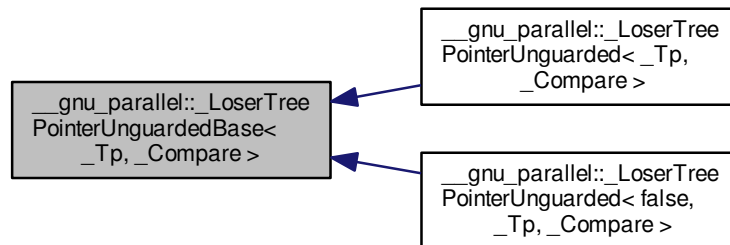
Definition at line 977 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.125 `__gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >`:



## Public Member Functions

- **`_LoserTreePointerUnguardedBase`** (unsigned int `__k`, const `_Tp` &`__sentinel`, `_Compare` `__comp`=`std::less<_Tp>()`)
- int **`__get_min_source`** ()
- void **`__insert_start`** (const `_Tp` &`__key`, int `__source`, bool)

## Protected Attributes

- `_Compare` **`_M_comp`**

- unsigned int `_M_ik`
- unsigned int `_M_k`
- `_Loser * _M_losers`
- unsigned int `_M_offset`

#### 4.125.1 Detailed Description

```
template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreePointerUnguardedBase<_Tp, _Compare>
```

Unguarded loser tree, keeping only pointers to the elements in the tree structure.

No guarding is done, therefore not a single input sequence must run empty. This is a very fast variant.

Definition at line 828 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 4.126 `__gnu_parallel::_LoserTreeTraits<_Tp>` Struct Template Reference

### Static Public Attributes

- static const bool `_M_use_pointer`

#### 4.126.1 Detailed Description

```
template<typename _Tp> struct __gnu_parallel::_LoserTreeTraits<_Tp>
```

Traits for determining whether the loser tree should use pointers or copies.

The field "`_M_use_pointer`" is used to determine whether to use pointers in the loser trees or whether to copy the values into the loser tree.

The default behavior is to use pointers if the data type is 4 times as big as the pointer to it.

Specialize for your data type to customize the behavior.

Example:

```
template<> struct _LoserTreeTraits<int> { static const bool _M_use_pointer = false; };
template<> struct _LoserTreeTraits<heavyweight_type> { static const bool _M_use_pointer = true; };
```

### Parameters

<code>_Tp</code>	type to give the loser tree traits for.
------------------	---

Definition at line 731 of file `multiway_merge.h`.

#### 4.126.2 Member Data Documentation

4.126.2.1 `template<typename _Tp> const bool __gnu_parallel::_LoserTreeTraits<_Tp>::_M_use_pointer` `[static]`

True iff to use pointers instead of values in loser trees.

The default behavior is to use pointers if the data type is four times as big as the pointer to it.



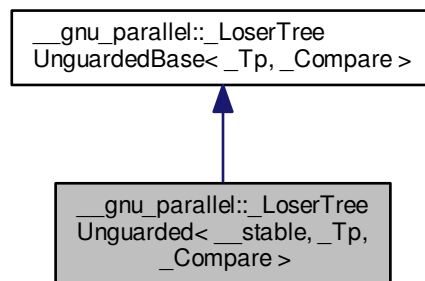
Definition at line 739 of file multiway\_merge.h.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

#### 4.127 `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >`:



##### Public Member Functions

- **\_LoserTreeUnguarded** (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=`std::less< _Tp >()`)
- void **\_\_delete\_min\_insert** (\_Tp \_\_key, bool)
- int **\_\_get\_min\_source** ()
- void **\_\_init** ()
- unsigned int **\_\_init\_winner** (unsigned int \_\_root)
- void **\_\_insert\_start** (const \_Tp &\_\_key, int \_\_source, bool)

##### Protected Attributes

- \_Compare **\_M\_comp**
- unsigned int **\_M\_ik**
- unsigned int **\_M\_k**
- \_Loser \* **\_M\_losers**
- unsigned int **\_M\_offset**

##### 4.127.1 Detailed Description

```
template<bool __stable, typename _Tp, typename _Compare>class __gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >
```

Stable implementation of unguarded `_LoserTree`.

Unstable variant is selected below with partial specialization.

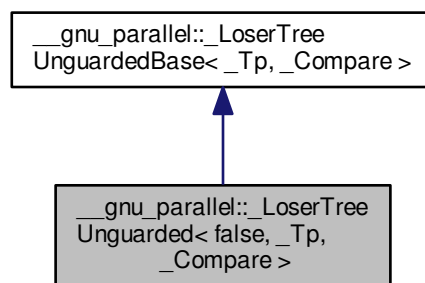
Definition at line 646 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

#### 4.128 `__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >`:



##### Public Member Functions

- **`_LoserTreeUnguarded`** (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=`std::less< _Tp >()`)
- void **`__delete_min_insert`** (\_Tp \_\_key, bool)
- int **`__get_min_source`** ()
- void **`__init`** ()
- unsigned int **`__init_winner`** (unsigned int \_\_root)
- void **`__insert_start`** (const \_Tp &\_\_key, int \_\_source, bool)

##### Protected Attributes

- \_Compare **`_M_comp`**
- unsigned int **`_M_ik`**
- unsigned int **`_M_k`**
- \_Loser \* **`_M_losers`**
- unsigned int **`_M_offset`**

##### 4.128.1 Detailed Description

```
template<typename _Tp, typename _Compare>class __gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >
```

Non-Stable implementation of unguarded `_LoserTree`.

Stable implementation is above.

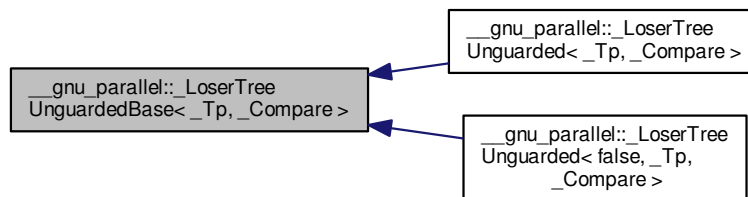
Definition at line 734 of file losertree.h.

The documentation for this class was generated from the following file:

- [losertree.h](#)

#### 4.129 `__gnu_parallel::_LoserTreeUnguardedBase<_Tp, _Compare>` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreeUnguardedBase<_Tp, _Compare>`:



##### Public Member Functions

- **`_LoserTreeUnguardedBase`** (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=`std::less<_Tp>()`)
- int **`__get_min_source`** ()
- void **`__insert_start`** (const \_Tp &\_\_key, int \_\_source, bool)

##### Protected Attributes

- `_Compare` **`_M_comp`**
- unsigned int **`_M_ik`**
- unsigned int **`_M_k`**
- `_Loser *` **`_M_losers`**
- unsigned int **`_M_offset`**

##### 4.129.1 Detailed Description

`template<typename _Tp, typename _Compare>class __gnu_parallel::_LoserTreeUnguardedBase<_Tp, _Compare>`

Base class for unguarded `_LoserTree` implementation.

The whole element is copied into the tree structure.

No guarding is done, therefore not a single input sequence must run empty. Unused `__sequence` heads are marked with a sentinel which is `>` all elements that are to be merged.

This is a very fast variant.

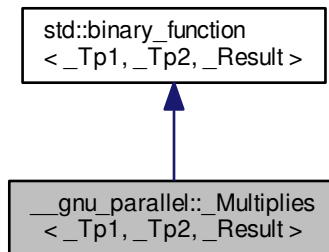
Definition at line 574 of file losertree.h.

The documentation for this class was generated from the following file:

- [losertree.h](#)

#### 4.130 `__gnu_parallel::_Multiplies<_Tp1, _Tp2, _Result>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::_Multiplies<_Tp1, _Tp2, _Result>`:



##### Public Types

- typedef `_Tp1` [first\\_argument\\_type](#)
- typedef `_Result` [result\\_type](#)
- typedef `_Tp2` [second\\_argument\\_type](#)

##### Public Member Functions

- `_Result` **operator()** (const `_Tp1` &`__x`, const `_Tp2` &`__y`) const

##### 4.130.1 Detailed Description

```
template<typename _Tp1, typename _Tp2, typename _Result = __typeof__(*static_cast<_Tp1*>(0) * *static_cast<_Tp2*>(0))> struct __gnu_parallel::_Multiplies<_Tp1, _Tp2, _Result>
```

Similar to `std::multiplies`, but allows two different types.

Definition at line 288 of file `parallel/base.h`.

##### 4.130.2 Member Typedef Documentation

4.130.2.1 typedef `_Tp1` `std::binary_function<_Tp1, _Tp2, _Result>::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

#### 4.130.2.2 `typedef _Result std::binary_function<_Tp1, _Tp2, _Result>::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

#### 4.130.2.3 `typedef _Tp2 std::binary_function<_Tp1, _Tp2, _Result>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

### 4.131 `__gnu_parallel::_Nothing` Struct Reference

#### Public Member Functions

- `template<typename _It> void operator() (_It __i)`

#### 4.131.1 Detailed Description

Functor doing nothing.

For some `__reduction` tasks (this is not a function object, but is passed as `__selector` `__dummy` parameter.

Definition at line 288 of file `for_each_selectors.h`.

#### 4.131.2 Member Function Documentation

##### 4.131.2.1 `template<typename _It> void __gnu_parallel::_Nothing::operator() (_It __i)` [inline]

Functor execution.

#### Parameters

<code>__i</code>	iterator referencing object.
------------------	------------------------------

Definition at line 294 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

### 4.132 `__gnu_parallel::_Piece<_DifferenceTp>` Struct Template Reference

#### Public Types

- `typedef _DifferenceTp _DifferenceType`

#### Public Attributes

- `_DifferenceType` `_M_begin`
- `_DifferenceType` `_M_end`

## 4.132.1 Detailed Description

```
template<typename _DifferenceTp>struct __gnu_parallel::_Piece<_DifferenceTp>
```

Subsequence description.

Definition at line 46 of file `multiway_mergesort.h`.

## 4.132.2 Member Data Documentation

4.132.2.1 `template<typename _DifferenceTp>_DifferenceType __gnu_parallel::_Piece<_DifferenceTp>::_M_begin`

Begin of subsequence.

Definition at line 51 of file `multiway_mergesort.h`.

4.132.2.2 `template<typename _DifferenceTp>_DifferenceType __gnu_parallel::_Piece<_DifferenceTp>::_M_end`

End of subsequence.

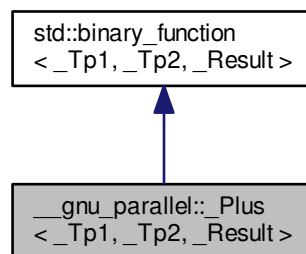
Definition at line 54 of file `multiway_mergesort.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

4.133 `__gnu_parallel::_Plus<_Tp1, _Tp2, _Result>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::_Plus<_Tp1, _Tp2, _Result>`:



## Public Types

- typedef `_Tp1` [first\\_argument\\_type](#)
- typedef `_Result` [result\\_type](#)
- typedef `_Tp2` [second\\_argument\\_type](#)

## Public Member Functions

- `_Result operator() (const _Tp1 &__x, const _Tp2 &__y) const`

### 4.133.1 Detailed Description

```
template<typename _Tp1, typename _Tp2, typename _Result = __typeof__(*static_cast<_Tp1*>(0) + *static_cast<_Tp2*>(0))> struct __gnu_parallel::Plus<_Tp1, _Tp2, _Result>
```

Similar to `std::plus`, but allows two different types.

Definition at line 272 of file `parallel/base.h`.

### 4.133.2 Member Typedef Documentation

4.133.2.1 `typedef _Tp1 std::binary_function<_Tp1, _Tp2, _Result>::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.133.2.2 `typedef _Result std::binary_function<_Tp1, _Tp2, _Result>::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.133.2.3 `typedef _Tp2 std::binary_function<_Tp1, _Tp2, _Result>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

## 4.134 `__gnu_parallel::PMWSSortingData<_RAIter>` Struct Template Reference

### Public Types

- `typedef _TraitsType::difference_type _DifferenceType`
- `typedef std::iterator_traits<_RAIter> _TraitsType`
- `typedef _TraitsType::value_type _ValueType`

### Public Attributes

- [\\_ThreadIndex](#) `_M_num_threads`
- `_DifferenceType` \* [\\_M\\_offsets](#)
- `std::vector<_Piece<_DifferenceType>>` \* [\\_M\\_pieces](#)
- `_ValueType` \* [\\_M\\_samples](#)
- `_RAIter` [\\_M\\_source](#)
- `_DifferenceType` \* [\\_M\\_starts](#)
- `_ValueType` \*\* [\\_M\\_temporary](#)

## 4.134.1 Detailed Description

`template<typename _RAIter> struct __gnu_parallel::PMWMSortingData<_RAIter>`

Data accessed by all threads.

PMWMS = parallel multiway mergesort

Definition at line 61 of file `multiway_mergesort.h`.

## 4.134.2 Member Data Documentation

4.134.2.1 `template<typename _RAIter> _ThreadIndex __gnu_parallel::PMWMSortingData<_RAIter>::M_num_threads`

Number of threads involved.

Definition at line 68 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

4.134.2.2 `template<typename _RAIter> _DifferenceType* __gnu_parallel::PMWMSortingData<_RAIter>::M_offsets`

Offsets to add to the found positions.

Definition at line 83 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`.

4.134.2.3 `template<typename _RAIter> std::vector<_Piece<_DifferenceType>>* __gnu_parallel::PMWMSortingData<_RAIter>::M_pieces`

Pieces of data to merge [`thread`][`sequence`].

Definition at line 86 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

4.134.2.4 `template<typename _RAIter> _ValueType* __gnu_parallel::PMWMSortingData<_RAIter>::M_samples`

Samples.

Definition at line 80 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::determine_samples()`, and `__gnu_parallel::parallel_sort_mwms()`.

4.134.2.5 `template<typename _RAIter> _RAIter __gnu_parallel::PMWMSortingData<_RAIter>::M_source`

Input `__begin`.

Definition at line 71 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::determine_samples()`, `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

4.134.2.6 `template<typename _RAIter> _DifferenceType* __gnu_parallel::PMWMSortingData<_RAIter>::M_starts`

Start indices, per thread.

Definition at line 74 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::determine_samples()`, `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.



::parallel\_sort\_mwms\_pu().

4.134.2.7 `template<typename _RAIter> _ValueType** __gnu_parallel::_PMWSSortingData<_RAIter>::_M_temporary`

Storage in which to sort.

Definition at line 77 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

## 4.135 `__gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>` Class Template Reference

### Public Types

- `typedef _DifferenceTp DifferenceType`
- `typedef \_PseudoSequenceIterator<_Tp, uint64_t> iterator`

### Public Member Functions

- [\\_PseudoSequence](#) (const \_Tp &\_\_val, \_DifferenceType \_\_count)
- [iterator begin](#) () const
- [iterator end](#) () const

### 4.135.1 Detailed Description

`template<typename _Tp, typename _DifferenceTp> class __gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>`

Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.

#### Parameters

<code>_Tp</code>	Sequence <code>_M_value</code> type.
<code>_DifferenceTp</code>	Sequence difference type.

Definition at line 359 of file `parallel/base.h`.

### 4.135.2 Constructor & Destructor Documentation

4.135.2.1 `template<typename _Tp, typename _DifferenceTp> __gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>::_PseudoSequence ( const _Tp &__val, _DifferenceType __count ) [inline]`

Constructor.

#### Parameters

<code>__val</code>	Element of the sequence.
--------------------	--------------------------

<code>__count</code>	Number of (virtual) copies.
----------------------	-----------------------------

Definition at line 371 of file `parallel/base.h`.

#### 4.135.3 Member Function Documentation

4.135.3.1 `template<typename _Tp, typename _DifferenceTp> iterator __gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>::begin ( ) const [inline]`

Begin iterator.

Definition at line 376 of file `parallel/base.h`.

4.135.3.2 `template<typename _Tp, typename _DifferenceTp> iterator __gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>::end ( ) const [inline]`

End iterator.

Definition at line 381 of file `parallel/base.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

## 4.136 `__gnu_parallel::_PseudoSequenceliterator<_Tp, _DifferenceTp>` Class Template Reference

### Public Types

- `typedef _DifferenceTp _DifferenceType`

### Public Member Functions

- `_PseudoSequenceliterator` (const `_Tp` &\_\_val, `_DifferenceType` \_\_pos)
- `bool operator!=` (const `_PseudoSequenceliterator` &\_\_i2)
- `const _Tp & operator*` ( ) const
- `_PseudoSequenceliterator & operator++` ( )
- `_PseudoSequenceliterator operator++` (int)
- `_DifferenceType operator-` (const `_PseudoSequenceliterator` &\_\_i2)
- `bool operator==` (const `_PseudoSequenceliterator` &\_\_i2)
- `const _Tp & operator[]` (`_DifferenceType`) const

#### 4.136.1 Detailed Description

`template<typename _Tp, typename _DifferenceTp> class __gnu_parallel::_PseudoSequenceliterator<_Tp, _DifferenceTp>`

`_Iterator` associated with `__gnu_parallel::_PseudoSequence`. It features the usual random-access iterator functionality.

### Parameters

<code>_Tp</code>	Sequence <code>_M_value</code> type.
<code>_DifferenceTp</code>	Sequence difference type.

Definition at line 306 of file `parallel/base.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

#### 4.137 `__gnu_parallel::QSBThreadLocal<_RAAlter>` Struct Template Reference

##### Public Types

- typedef `_TraitsType::difference_type` **`_DifferenceType`**
- typedef `std::pair<_RAAlter, _RAAlter>` `_Piece`
- typedef `std::iterator_traits<_RAAlter>` **`_TraitsType`**

##### Public Member Functions

- `_QSBThreadLocal` (int `__queue_size`)

##### Public Attributes

- volatile `_DifferenceType` \* `_M_elements_leftover`
- `_Piece` `_M_global`
- `_Piece` `_M_initial`
- `_RestrictedBoundedConcurrentQueue<_Piece>` `_M_leftover_parts`
- `_ThreadIndex` `_M_num_threads`

##### 4.137.1 Detailed Description

```
template<typename _RAAlter> struct __gnu_parallel::QSBThreadLocal<_RAAlter>
```

Information local to one thread in the parallel quicksort run.

Definition at line 62 of file `balanced_quicksort.h`.

##### 4.137.2 Member Typedef Documentation

4.137.2.1 `template<typename _RAAlter> typedef std::pair<_RAAlter, _RAAlter> __gnu_parallel::QSBThreadLocal<_RAAlter>::_Piece`

Continuous part of the sequence, described by an iterator pair.

Definition at line 69 of file `balanced_quicksort.h`.

##### 4.137.3 Constructor & Destructor Documentation

4.137.3.1 `template<typename _RAAlter> __gnu_parallel::QSBThreadLocal<_RAAlter>::QSBThreadLocal ( int __queue_size ) [inline]`

Constructor.

## Parameters

<code>__queue_size</code>	size of the work-stealing queue.
---------------------------	----------------------------------

Definition at line 88 of file `balanced_quicksort.h`.

## 4.137.4 Member Data Documentation

4.137.4.1 `template<typename _RAIter> volatile _DifferenceType* __gnu_parallel::_QSBThreadLocal<_RAIter>::__M_elements_leftover`

Pointer to a counter of elements left over to sort.

Definition at line 81 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::_parallel_sort_qsb()`, `__gnu_parallel::_qsb_conquer()`, and `__gnu_parallel::_qsb_local_sort_with_helping()`.

4.137.4.2 `template<typename _RAIter> _Piece __gnu_parallel::_QSBThreadLocal<_RAIter>::__M_global`

The complete sequence to sort.

Definition at line 84 of file `balanced_quicksort.h`.

4.137.4.3 `template<typename _RAIter> _Piece __gnu_parallel::_QSBThreadLocal<_RAIter>::__M_initial`

Initial piece to work on.

Definition at line 72 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::_qsb_conquer()`, and `__gnu_parallel::_qsb_local_sort_with_helping()`.

4.137.4.4 `template<typename _RAIter> _RestrictedBoundedConcurrentQueue<_Piece> __gnu_parallel::_QSBThreadLocal<_RAIter>::__M_leftover_parts`

Work-stealing queue.

Definition at line 75 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::_qsb_local_sort_with_helping()`.

4.137.4.5 `template<typename _RAIter> _ThreadIndex __gnu_parallel::_QSBThreadLocal<_RAIter>::__M_num_threads`

Number of threads involved in this algorithm.

Definition at line 78 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::_qsb_local_sort_with_helping()`.

The documentation for this struct was generated from the following file:

- [balanced\\_quicksort.h](#)

4.138 `__gnu_parallel::_RandomNumber` Class Reference

## Public Member Functions

- [\\_RandomNumber](#) ()
- [\\_RandomNumber](#) (uint32\_t \_\_seed, uint64\_t \_\_M\_supremum=0x100000000ULL)
- unsigned long [\\_\\_genrand\\_bits](#) (int \_\_bits)

- `uint32_t operator() ()`
- `uint32_t operator() (uint64_t local_supremum)`

#### 4.138.1 Detailed Description

Random number generator, based on the Mersenne twister.

Definition at line 42 of file `random_number.h`.

#### 4.138.2 Constructor & Destructor Documentation

##### 4.138.2.1 `__gnu_parallel::RandomNumber::RandomNumber ( ) [inline]`

Default constructor. Seed with 0.

Definition at line 74 of file `random_number.h`.

##### 4.138.2.2 `__gnu_parallel::RandomNumber::RandomNumber ( uint32_t __seed, uint64_t _M_supremum = 0x100000000ULL ) [inline]`

Constructor.

Parameters

<code>__seed</code>	Random __seed.
<code>_M_supremum</code>	Generate integer random numbers in the interval [0,_M_supremum).

Definition at line 85 of file `random_number.h`.

#### 4.138.3 Member Function Documentation

##### 4.138.3.1 `unsigned long __gnu_parallel::RandomNumber::genrand_bits ( int __bits ) [inline]`

Generate a number of random bits, run-time parameter.

Parameters

<code>__bits</code>	Number of bits to generate.
---------------------	-----------------------------

Definition at line 109 of file `random_number.h`.

##### 4.138.3.2 `uint32_t __gnu_parallel::RandomNumber::operator() ( ) [inline]`

Generate unsigned random 32-bit integer.

Definition at line 94 of file `random_number.h`.

##### 4.138.3.3 `uint32_t __gnu_parallel::RandomNumber::operator() ( uint64_t local_supremum ) [inline]`

Generate unsigned random 32-bit integer in the interval [0,local\_supremum).

Definition at line 100 of file `random_number.h`.

The documentation for this class was generated from the following file:

- [random\\_number.h](#)

4.139 `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>` Class Template Reference

## Public Member Functions

- `_RestrictedBoundedConcurrentQueue ( _SequenceIndex __max_size )`
- `~_RestrictedBoundedConcurrentQueue ( )`
- `bool pop_back ( _Tp &__t )`
- `bool pop_front ( _Tp &__t )`
- `void push_front ( const _Tp &__t )`

## 4.139.1 Detailed Description

```
template<typename _Tp>class __gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>
```

Double-ended queue of bounded size, allowing lock-free atomic access. `push_front()` and `pop_front()` must not be called concurrently to each other, while `pop_back()` can be called concurrently at all times. `empty()`, `size()`, and `top()` are intentionally not provided. Calling them would not make sense in a concurrent setting.

## Parameters

<code>_Tp</code>	Contained element type.
------------------	-------------------------

Definition at line 52 of file `queue.h`.

## 4.139.2 Constructor &amp; Destructor Documentation

```
4.139.2.1 template<typename _Tp> __gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>
::_RestrictedBoundedConcurrentQueue ( _SequenceIndex __max_size ) [inline]
```

Constructor. Not to be called concurrent, of course.

## Parameters

<code>__max_size</code>	Maximal number of elements to be contained.
-------------------------	---

Definition at line 68 of file `queue.h`.

```
4.139.2.2 template<typename _Tp> __gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>
::~~_RestrictedBoundedConcurrentQueue ( ) [inline]
```

Destructor. Not to be called concurrent, of course.

Definition at line 77 of file `queue.h`.

## 4.139.3 Member Function Documentation

```
4.139.3.1 template<typename _Tp> bool __gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>::pop_back (
_Tp &__t ) [inline]
```

Pops one element from the queue at the front end. Must not be called concurrently with `pop_front()`.

Definition at line 127 of file `queue.h`.

4.139.3.2 `template<typename _Tp> bool __gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>::pop_front ( _Tp &__t ) [inline]`

Pops one element from the queue at the front end. Must not be called concurrently with `pop_front()`.

Definition at line 100 of file `queue.h`.

4.139.3.3 `template<typename _Tp> void __gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>::push_front ( const _Tp &__t ) [inline]`

Pushes one element into the queue at the front end. Must not be called concurrently with `pop_front()`.

Definition at line 83 of file `queue.h`.

The documentation for this class was generated from the following file:

- [queue.h](#)

#### 4.140 `__gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >` Struct Template Reference

##### Public Member Functions

- void **operator()** ( \_RAIter \_\_first, \_RAIter \_\_last, \_StrictWeakOrdering \_\_comp)

##### 4.140.1 Detailed Description

`template<bool __stable, class _RAIter, class _StrictWeakOrdering> struct __gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >`

Stable sorting functor.

Used to reduce code instantiation in `multiway_merge_sampling_splitting`.

Definition at line 1007 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

#### 4.141 `__gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >` Struct Template Reference

##### Public Member Functions

- void **operator()** ( \_RAIter \_\_first, \_RAIter \_\_last, \_StrictWeakOrdering \_\_comp)

##### 4.141.1 Detailed Description

`template<class _RAIter, class _StrictWeakOrdering> struct __gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >`

Non-`__stable` sorting functor.

Used to reduce code instantiation in `multiway_merge_sampling_splitting`.

Definition at line 1020 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

## 4.142 \_\_gnu\_parallel::\_Settings Struct Reference

### Static Public Member Functions

- static const [\\_Settings](#) & [get](#) () throw ()
- static void [set](#) ([\\_Settings](#) &) throw ()

### Public Attributes

- [\\_SequenceIndex](#) [accumulate\\_minimal\\_n](#)
- unsigned int [adjacent\\_difference\\_minimal\\_n](#)
- [\\_AlgorithmStrategy](#) [algorithm\\_strategy](#)
- unsigned int [cache\\_line\\_size](#)
- [\\_SequenceIndex](#) [count\\_minimal\\_n](#)
- [\\_SequenceIndex](#) [fill\\_minimal\\_n](#)
- [\\_FindAlgorithm](#) [find\\_algorithm](#)
- double [find\\_increasing\\_factor](#)
- [\\_SequenceIndex](#) [find\\_initial\\_block\\_size](#)
- [\\_SequenceIndex](#) [find\\_maximum\\_block\\_size](#)
- float [find\\_scale\\_factor](#)
- [\\_SequenceIndex](#) [find\\_sequential\\_search\\_size](#)
- [\\_SequenceIndex](#) [for\\_each\\_minimal\\_n](#)
- [\\_SequenceIndex](#) [generate\\_minimal\\_n](#)
- unsigned long long [L1\\_cache\\_size](#)
- unsigned long long [L2\\_cache\\_size](#)
- [\\_SequenceIndex](#) [max\\_element\\_minimal\\_n](#)
- [\\_SequenceIndex](#) [merge\\_minimal\\_n](#)
- unsigned int [merge\\_oversampling](#)
- [\\_SplittingAlgorithm](#) [merge\\_splitting](#)
- [\\_SequenceIndex](#) [min\\_element\\_minimal\\_n](#)
- [\\_MultiwayMergeAlgorithm](#) [multiway\\_merge\\_algorithm](#)
- int [multiway\\_merge\\_minimal\\_k](#)
- [\\_SequenceIndex](#) [multiway\\_merge\\_minimal\\_n](#)
- unsigned int [multiway\\_merge\\_oversampling](#)
- [\\_SplittingAlgorithm](#) [multiway\\_merge\\_splitting](#)
- [\\_SequenceIndex](#) [nth\\_element\\_minimal\\_n](#)
- [\\_SequenceIndex](#) [partial\\_sort\\_minimal\\_n](#)
- [\\_PartialSumAlgorithm](#) [partial\\_sum\\_algorithm](#)
- float [partial\\_sum\\_dilation](#)
- unsigned int [partial\\_sum\\_minimal\\_n](#)
- double [partition\\_chunk\\_share](#)
- [\\_SequenceIndex](#) [partition\\_chunk\\_size](#)
- [\\_SequenceIndex](#) [partition\\_minimal\\_n](#)
- [\\_SequenceIndex](#) [qsb\\_steals](#)
- unsigned int [random\\_shuffle\\_minimal\\_n](#)
- [\\_SequenceIndex](#) [replace\\_minimal\\_n](#)
- [\\_SequenceIndex](#) [search\\_minimal\\_n](#)
- [\\_SequenceIndex](#) [set\\_difference\\_minimal\\_n](#)



- [\\_SequenceIndex set\\_intersection\\_minimal\\_n](#)
- [\\_SequenceIndex set\\_symmetric\\_difference\\_minimal\\_n](#)
- [\\_SequenceIndex set\\_union\\_minimal\\_n](#)
- [\\_SortAlgorithm sort\\_algorithm](#)
- [\\_SequenceIndex sort\\_minimal\\_n](#)
- unsigned int [sort\\_mwms\\_oversampling](#)
- unsigned int [sort\\_qs\\_num\\_samples\\_preset](#)
- [\\_SequenceIndex sort\\_qsb\\_base\\_case\\_maximal\\_n](#)
- [\\_SplittingAlgorithm sort\\_splitting](#)
- unsigned int [TLB\\_size](#)
- [\\_SequenceIndex transform\\_minimal\\_n](#)
- [\\_SequenceIndex unique\\_copy\\_minimal\\_n](#)
- [\\_SequenceIndex workstealing\\_chunk\\_size](#)

#### 4.142.1 Detailed Description

class `_Settings` Run-time settings for the parallel mode including all tunable parameters.

Definition at line 123 of file `settings.h`.

#### 4.142.2 Member Function Documentation

##### 4.142.2.1 static const `_Settings& __gnu_parallel::_Settings::get( ) throw` [static]

Get the global settings.

Referenced by `__gnu_parallel::_find_template()`, `__gnu_parallel::_for_each_template_random_access_workstealing()`, `__gnu_parallel::_parallel_nth_element()`, `__gnu_parallel::_parallel_partial_sum()`, `__gnu_parallel::_parallel_partial_sum_linear()`, `__gnu_parallel::_parallel_partition()`, `__gnu_parallel::_parallel_random_shuffle_drs()`, `__gnu_parallel::_parallel_sort()`, `__gnu_parallel::_parallel_sort_qs_conquer()`, `__gnu_parallel::_qsb_local_sort_with_helping()`, `__gnu_parallel::_sequential_random_shuffle()`, `__gnu_parallel::_multiway_merge_sampling_splitting()`, `__gnu_parallel::_parallel_multiway_merge()`, `__gnu_parallel::_parallel_sort_mwms()`, and `__gnu_parallel::_parallel_sort_mwms_pu()`.

##### 4.142.2.2 static void `__gnu_parallel::_Settings::set( _Settings & ) throw` [static]

Set the global settings.

#### 4.142.3 Member Data Documentation

##### 4.142.3.1 `_SequenceIndex __gnu_parallel::_Settings::accumulate_minimal_n`

Minimal input size for `accumulate`.

Definition at line 139 of file `settings.h`.

##### 4.142.3.2 unsigned int `__gnu_parallel::_Settings::adjacent_difference_minimal_n`

Minimal input size for `adjacent_difference`.

Definition at line 142 of file `settings.h`.

#### 4.142.3.3 unsigned int \_\_gnu\_parallel::\_Settings::cache\_line\_size

Overestimation of cache line size. Used to avoid false sharing, i.e. elements of different threads are at least this amount apart.

Definition at line 265 of file settings.h.

Referenced by \_\_gnu\_parallel::\_\_for\_each\_template\_random\_access\_workstealing().

#### 4.142.3.4 \_SequenceIndex \_\_gnu\_parallel::\_Settings::count\_minimal\_n

Minimal input size for count and count\_if.

Definition at line 145 of file settings.h.

#### 4.142.3.5 \_SequenceIndex \_\_gnu\_parallel::\_Settings::fill\_minimal\_n

Minimal input size for fill.

Definition at line 148 of file settings.h.

#### 4.142.3.6 double \_\_gnu\_parallel::\_Settings::find\_increasing\_factor

Block size increase factor for find.

Definition at line 151 of file settings.h.

#### 4.142.3.7 \_SequenceIndex \_\_gnu\_parallel::\_Settings::find\_initial\_block\_size

Initial block size for find.

Definition at line 154 of file settings.h.

Referenced by \_\_gnu\_parallel::\_\_find\_template().

#### 4.142.3.8 \_SequenceIndex \_\_gnu\_parallel::\_Settings::find\_maximum\_block\_size

Maximal block size for find.

Definition at line 157 of file settings.h.

#### 4.142.3.9 float \_\_gnu\_parallel::\_Settings::find\_scale\_factor

Block size scale-down factor with respect to current position.

Definition at line 276 of file settings.h.

Referenced by \_\_gnu\_parallel::\_\_find\_template().

#### 4.142.3.10 \_SequenceIndex \_\_gnu\_parallel::\_Settings::find\_sequential\_search\_size

Start with looking for this many elements sequentially, for find.

Definition at line 160 of file settings.h.

Referenced by \_\_gnu\_parallel::\_\_find\_template().

#### 4.142.3.11 \_SequenceIndex \_\_gnu\_parallel::\_Settings::for\_each\_minimal\_n

Minimal input size for for\_each.

Definition at line 163 of file settings.h.

**4.142.3.12 \_\_SequenceIndex \_\_gnu\_parallel::\_Settings::generate\_minimal\_n**

Minimal input size for generate.

Definition at line 166 of file settings.h.

**4.142.3.13 unsigned long long \_\_gnu\_parallel::\_Settings::L1\_cache\_size**

size of the L1 cache in bytes (underestimation).

Definition at line 254 of file settings.h.

**4.142.3.14 unsigned long long \_\_gnu\_parallel::\_Settings::L2\_cache\_size**

size of the L2 cache in bytes (underestimation).

Definition at line 257 of file settings.h.

Referenced by `__gnu_parallel::_parallel_random_shuffle_drs()`, and `__gnu_parallel::_sequential_random_shuffle()`.

**4.142.3.15 \_\_SequenceIndex \_\_gnu\_parallel::\_Settings::max\_element\_minimal\_n**

Minimal input size for `max_element`.

Definition at line 169 of file settings.h.

**4.142.3.16 \_\_SequenceIndex \_\_gnu\_parallel::\_Settings::merge\_minimal\_n**

Minimal input size for `merge`.

Definition at line 172 of file settings.h.

**4.142.3.17 unsigned int \_\_gnu\_parallel::\_Settings::merge\_oversampling**

Oversampling factor for `merge`.

Definition at line 175 of file settings.h.

Referenced by `__gnu_parallel::_multiway_merge_sampling_splitting()`, and `__gnu_parallel::_parallel_multiway_merge()`.

**4.142.3.18 \_\_SequenceIndex \_\_gnu\_parallel::\_Settings::min\_element\_minimal\_n**

Minimal input size for `min_element`.

Definition at line 178 of file settings.h.

**4.142.3.19 int \_\_gnu\_parallel::\_Settings::multiway\_merge\_minimal\_k**

Oversampling factor for `multiway_merge`.

Definition at line 184 of file settings.h.

**4.142.3.20 \_\_SequenceIndex \_\_gnu\_parallel::\_Settings::multiway\_merge\_minimal\_n**

Minimal input size for `multiway_merge`.

Definition at line 181 of file settings.h.

**4.142.3.21 unsigned int \_\_gnu\_parallel::\_Settings::multiway\_merge\_oversampling**

Oversampling factor for `multiway_merge`.

Definition at line 187 of file settings.h.

**4.142.3.22 \_\_SequenceIndex \_\_gnu\_parallel::\_Settings::nth\_element\_minimal\_n**

Minimal input size for nth\_element.

Definition at line 190 of file settings.h.

Referenced by \_\_gnu\_parallel::\_\_parallel\_nth\_element().

**4.142.3.23 \_\_SequenceIndex \_\_gnu\_parallel::\_Settings::partial\_sort\_minimal\_n**

Minimal input size for partial\_sort.

Definition at line 203 of file settings.h.

**4.142.3.24 float \_\_gnu\_parallel::\_Settings::partial\_sum\_dilation**

Ratio for partial\_sum. Assume "sum and write result" to be this factor slower than just "sum".

Definition at line 207 of file settings.h.

Referenced by \_\_gnu\_parallel::\_\_parallel\_partial\_sum\_linear().

**4.142.3.25 unsigned int \_\_gnu\_parallel::\_Settings::partial\_sum\_minimal\_n**

Minimal input size for partial\_sum.

Definition at line 210 of file settings.h.

**4.142.3.26 double \_\_gnu\_parallel::\_Settings::partition\_chunk\_share**

Chunk size for partition, relative to input size. If > 0.0, this value overrides partition\_chunk\_size.

Definition at line 197 of file settings.h.

Referenced by \_\_gnu\_parallel::\_\_parallel\_partition().

**4.142.3.27 \_\_SequenceIndex \_\_gnu\_parallel::\_Settings::partition\_chunk\_size**

Chunk size for partition.

Definition at line 193 of file settings.h.

Referenced by \_\_gnu\_parallel::\_\_parallel\_partition().

**4.142.3.28 \_\_SequenceIndex \_\_gnu\_parallel::\_Settings::partition\_minimal\_n**

Minimal input size for partition.

Definition at line 200 of file settings.h.

Referenced by \_\_gnu\_parallel::\_\_parallel\_nth\_element().

**4.142.3.29 \_\_SequenceIndex \_\_gnu\_parallel::\_Settings::qsb\_steals**

The number of stolen ranges in load-balanced quicksort.

Definition at line 270 of file settings.h.

**4.142.3.30 unsigned int \_\_gnu\_parallel::\_Settings::random\_shuffle\_minimal\_n**

Minimal input size for random\_shuffle.

Definition at line 213 of file settings.h.

#### 4.142.3.31 `_SequenceIndex __gnu_parallel::_Settings::replace_minimal_n`

Minimal input size for `replace` and `replace_if`.

Definition at line 216 of file `settings.h`.

#### 4.142.3.32 `_SequenceIndex __gnu_parallel::_Settings::search_minimal_n`

Minimal input size for `search` and `search_n`.

Definition at line 273 of file `settings.h`.

#### 4.142.3.33 `_SequenceIndex __gnu_parallel::_Settings::set_difference_minimal_n`

Minimal input size for `set_difference`.

Definition at line 219 of file `settings.h`.

#### 4.142.3.34 `_SequenceIndex __gnu_parallel::_Settings::set_intersection_minimal_n`

Minimal input size for `set_intersection`.

Definition at line 222 of file `settings.h`.

#### 4.142.3.35 `_SequenceIndex __gnu_parallel::_Settings::set_symmetric_difference_minimal_n`

Minimal input size for `set_symmetric_difference`.

Definition at line 225 of file `settings.h`.

#### 4.142.3.36 `_SequenceIndex __gnu_parallel::_Settings::set_union_minimal_n`

Minimal input size for `set_union`.

Definition at line 228 of file `settings.h`.

#### 4.142.3.37 `_SequenceIndex __gnu_parallel::_Settings::sort_minimal_n`

Minimal input size for parallel sorting.

Definition at line 231 of file `settings.h`.

#### 4.142.3.38 `unsigned int __gnu_parallel::_Settings::sort_mwms_oversampling`

Oversampling factor for parallel `std::sort` (MWMS).

Definition at line 234 of file `settings.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

#### 4.142.3.39 `unsigned int __gnu_parallel::_Settings::sort_qs_num_samples_preset`

Such many samples to take to find a good pivot (quicksort).

Definition at line 237 of file `settings.h`.

#### 4.142.3.40 `_SequenceIndex __gnu_parallel::_Settings::sort_qsb_base_case_maximal_n`

Maximal subsequence `__length` to switch to unbalanced `__base` case. Applies to `std::sort` with dynamically load-balanced quicksort.

Definition at line 241 of file `settings.h`.

Referenced by `__gnu_parallel::_qsb_local_sort_with_helping()`.

4.142.3.41 `unsigned int __gnu_parallel::_Settings::TLB_size`

size of the Translation Lookaside Buffer (underestimation).

Definition at line 260 of file `settings.h`.

Referenced by `__gnu_parallel::_parallel_random_shuffle_drs()`, and `__gnu_parallel::_sequential_random_shuffle()`.

4.142.3.42 `_SequenceIndex __gnu_parallel::_Settings::transform_minimal_n`

Minimal input size for `parallel std::transform`.

Definition at line 244 of file `settings.h`.

4.142.3.43 `_SequenceIndex __gnu_parallel::_Settings::unique_copy_minimal_n`

Minimal input size for `unique_copy`.

Definition at line 247 of file `settings.h`.

The documentation for this struct was generated from the following file:

- [settings.h](#)

4.143 `__gnu_parallel::_SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator > Struct` Template Reference

4.143.1 Detailed Description

```
template<bool __exact, typename _RAIter, typename _Compare, typename _SortingPlacesIterator>struct __gnu_parallel::_SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator >
```

Split consistently.

Definition at line 122 of file `multiway_mergesort.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

4.144 `__gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator > Struct` Template Reference

Public Member Functions

- void **operator()** (const [\\_ThreadIndex](#) \_\_iam, [\\_PMWSSortingData](#)< \_RAIter > \*\_\_sd, \_Compare &\_\_comp, const typename `std::iterator_traits< _RAIter >::difference_type` \_\_num\_samples) const

4.144.1 Detailed Description

```
template<typename _RAIter, typename _Compare, typename _SortingPlacesIterator>struct __gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >
```

Split by sampling.

Definition at line 187 of file multiway\_mergesort.h.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

#### 4.145 `__gnu_parallel::_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >` Struct Template Reference

##### Public Member Functions

- void **operator()** (const [\\_ThreadIndex](#) \_\_iam, [\\_PMWSSortingData](#)< [\\_RAIter](#) > \*\_\_sd, [\\_Compare](#) &\_\_comp, const typename std::iterator\_traits< [\\_RAIter](#) >::difference\_type \_\_num\_samples) const

##### 4.145.1 Detailed Description

```
template<typename _RAIter, typename _Compare, typename _SortingPlacesIterator> struct __gnu_parallel::_SplitConsistently< true,
_RAlter, _Compare, _SortingPlacesIterator >
```

Split by exact splitting.

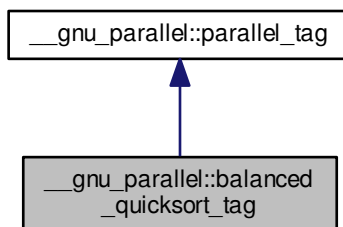
Definition at line 128 of file multiway\_mergesort.h.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

#### 4.146 `__gnu_parallel::balanced_quicksort_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::balanced_quicksort_tag`:



##### Public Member Functions

- **balanced\_quicksort\_tag** ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) **\_\_get\_num\_threads** ()
- void **set\_num\_threads** ([\\_ThreadIndex](#) \_\_num\_threads)

## 4.146.1 Detailed Description

Forces parallel sorting using balanced quicksort at compile time.

Definition at line 164 of file tags.h.

## 4.146.2 Member Function Documentation

4.146.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::get_num_threads ( )` `[inline]`, `[inherited]`

Find out desired number of threads.

## Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

4.146.2.2 `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex __num_threads )` `[inline]`, `[inherited]`

Set the desired number of threads.

## Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

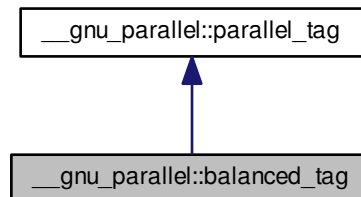
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.147 `__gnu_parallel::balanced_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::balanced_tag`:



## Public Member Functions

- [\\_ThreadIndex \\_\\_get\\_num\\_threads \( \)](#)



- void [set\\_num\\_threads](#) ([\\_ThreadIndex](#) \_\_num\_threads)

#### 4.147.1 Detailed Description

Recommends parallel execution using dynamic load-balancing at compile time.

Definition at line 88 of file tags.h.

#### 4.147.2 Member Function Documentation

##### 4.147.2.1 [\\_ThreadIndex](#) [\\_\\_gnu\\_parallel::parallel\\_tag::get\\_num\\_threads](#) ( ) [\[inline\]](#), [\[inherited\]](#)

Find out desired number of threads.

##### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort](#)().

##### 4.147.2.2 void [\\_\\_gnu\\_parallel::parallel\\_tag::set\\_num\\_threads](#) ( [\\_ThreadIndex](#) [\\_\\_num\\_threads](#) ) [\[inline\]](#), [\[inherited\]](#)

Set the desired number of threads.

##### Parameters

<a href="#">__num_threads</a>	Desired number of threads.
-------------------------------	----------------------------

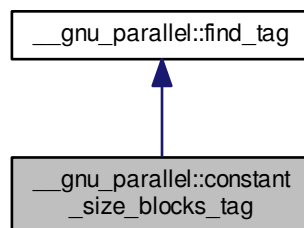
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

#### 4.148 [\\_\\_gnu\\_parallel::constant\\_size\\_blocks\\_tag](#) Struct Reference

Inheritance diagram for [\\_\\_gnu\\_parallel::constant\\_size\\_blocks\\_tag](#):



## 4.148.1 Detailed Description

Selects the constant block size variant for `std::find()`.

See also

`_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`

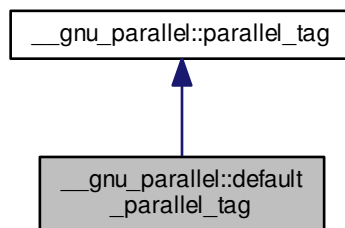
Definition at line 178 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.149 `__gnu_parallel::default_parallel_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::default_parallel_tag`:



## Public Member Functions

- **`default_parallel_tag`** (`_ThreadIndex` `__num_threads`)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` `__num_threads`)

## 4.149.1 Detailed Description

Recommends parallel execution using the default parallel algorithm.

Definition at line 79 of file `tags.h`.

## 4.149.2 Member Function Documentation

4.149.2.1 `_ThreadIndex` `__gnu_parallel::parallel_tag::__get_num_threads` ( ) `[inline]`, `[inherited]`

Find out desired number of threads.

**Returns**

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

4.149.2.2 `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex __num_threads ) [inline], [inherited]`

Set the desired number of threads.

**Parameters**

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

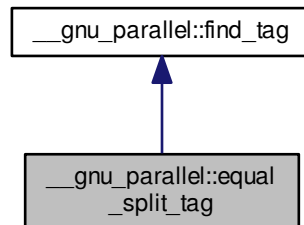
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

**4.150 \_\_gnu\_parallel::equal\_split\_tag Struct Reference**

Inheritance diagram for `__gnu_parallel::equal_split_tag`:

**4.150.1 Detailed Description**

Selects the equal splitting variant for `std::find()`.

**See also**

`_GLIBCXX_FIND_EQUAL_SPLIT`

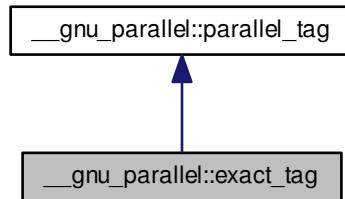
Definition at line 182 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.151 `__gnu_parallel::exact_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::exact_tag`:



## Public Member Functions

- **`exact_tag`** (`_ThreadIndex` `__num_threads`)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` `__num_threads`)

## 4.151.1 Detailed Description

Forces parallel merging with exact splitting, at compile time.

Definition at line 109 of file `tags.h`.

## 4.151.2 Member Function Documentation

4.151.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads` ( ) `[inline]`, `[inherited]`

Find out desired number of threads.

## Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::__parallel_sort`().

4.151.2.2 void `__gnu_parallel::parallel_tag::set_num_threads` ( `_ThreadIndex` `__num_threads` ) `[inline]`, `[inherited]`

Set the desired number of threads.

#### Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

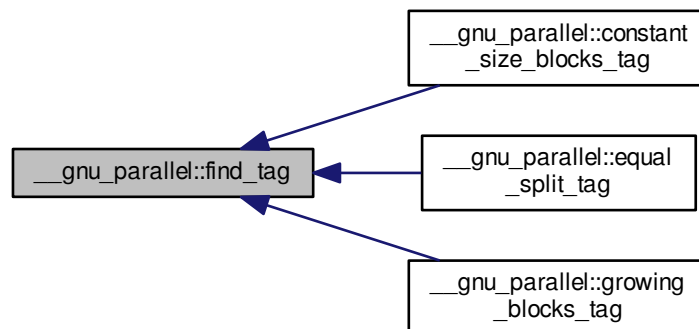
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

### 4.152 `__gnu_parallel::find_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::find_tag`:



#### 4.152.1 Detailed Description

Base class for `std::find()` variants.

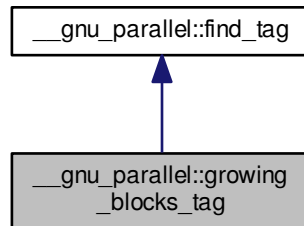
Definition at line 104 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.153 `__gnu_parallel::growing_blocks_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::growing_blocks_tag`:



## 4.153.1 Detailed Description

Selects the growing block size variant for `std::find()`.

See also

`_GLIBCXX_FIND_GROWING_BLOCKS`

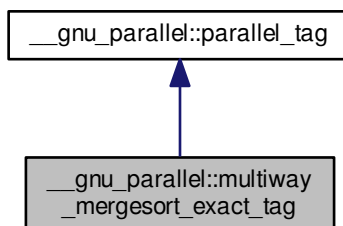
Definition at line 174 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.154 `__gnu_parallel::multiway_mergesort_exact_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::multiway_mergesort_exact_tag`:



## Public Member Functions

- **multiway\_mergesort\_exact\_tag** ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) **\_\_get\_num\_threads** ()
- void **set\_num\_threads** ([\\_ThreadIndex](#) \_\_num\_threads)

## 4.154.1 Detailed Description

Forces parallel sorting using multiway mergesort with exact splitting at compile time.

Definition at line 137 of file tags.h.

## 4.154.2 Member Function Documentation

4.154.2.1 [\\_ThreadIndex](#) **\_\_gnu\_parallel::parallel\_tag::\_\_get\_num\_threads** ( ) [\[inline\]](#), [\[inherited\]](#)

Find out desired number of threads.

## Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#).

4.154.2.2 void **\_\_gnu\_parallel::parallel\_tag::set\_num\_threads** ( [\\_ThreadIndex](#) *\_\_num\_threads* ) [\[inline\]](#), [\[inherited\]](#)

Set the desired number of threads.

## Parameters

<a href="#">__num_threads</a>	Desired number of threads.
-------------------------------	----------------------------

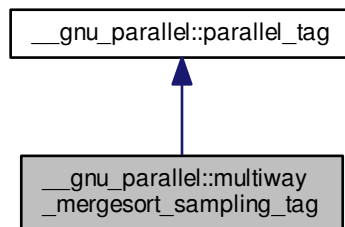
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.155 `__gnu_parallel::multiway_mergesort_sampling_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::multiway_mergesort_sampling_tag`:



## Public Member Functions

- `multiway_mergesort_sampling_tag` ( `_ThreadIndex` `__num_threads` )
- `_ThreadIndex` `__get_num_threads` ( )
- void `set_num_threads` ( `_ThreadIndex` `__num_threads` )

## 4.155.1 Detailed Description

Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.

Definition at line 146 of file `tags.h`.

## 4.155.2 Member Function Documentation

4.155.2.1 `_ThreadIndex` `__gnu_parallel::parallel_tag::__get_num_threads` ( ) `[inline]`, `[inherited]`

Find out desired number of threads.

## Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::__parallel_sort`( ).

4.155.2.2 void `__gnu_parallel::parallel_tag::set_num_threads` ( `_ThreadIndex` `__num_threads` ) `[inline]`, `[inherited]`

Set the desired number of threads.



## Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

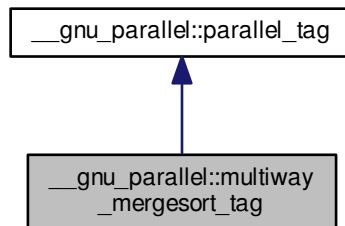
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

#### 4.156 `__gnu_parallel::multiway_mergesort_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::multiway_mergesort_tag`:



## Public Member Functions

- **`multiway_mergesort_tag`** (`_ThreadIndex` `__num_threads`)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` `__num_threads`)

##### 4.156.1 Detailed Description

Forces parallel sorting using multiway mergesort at compile time.

Definition at line 128 of file tags.h.

##### 4.156.2 Member Function Documentation

###### 4.156.2.1 `_ThreadIndex` `__gnu_parallel::parallel_tag::__get_num_threads` ( ) `[inline]`, `[inherited]`

Find out desired number of threads.

## Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort`().

4.156.2.2 `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex __num_threads )` [inline],  
[inherited]

Set the desired number of threads.

## Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

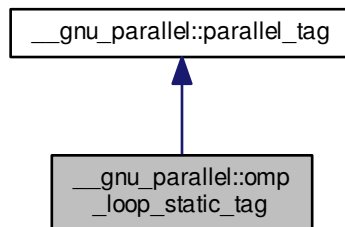
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

#### 4.157 `__gnu_parallel::omp_loop_static_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::omp_loop_static_tag`:



## Public Member Functions

- [\\_ThreadIndex \\_\\_get\\_num\\_threads\(\)](#)
- void [set\\_num\\_threads\(\\_ThreadIndex \\_\\_num\\_threads\)](#)

##### 4.157.1 Detailed Description

Recommends parallel execution using OpenMP static load-balancing at compile time.

Definition at line 100 of file tags.h.

##### 4.157.2 Member Function Documentation

###### 4.157.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads()` `[inline]`, `[inherited]`

Find out desired number of threads.

## Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

4.157.2.2 `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex __num_threads )` [inline],  
[inherited]

Set the desired number of threads.

## Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

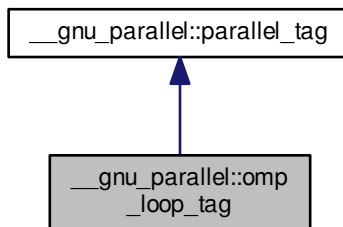
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

#### 4.158 `__gnu_parallel::omp_loop_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::omp_loop_tag`:



## Public Member Functions

- [\\_ThreadIndex \\_\\_get\\_num\\_threads\(\)](#)
- [void set\\_num\\_threads\(\\_ThreadIndex \\_\\_num\\_threads\)](#)

##### 4.158.1 Detailed Description

Recommends parallel execution using OpenMP dynamic load-balancing at compile time.

Definition at line 96 of file tags.h.

##### 4.158.2 Member Function Documentation

###### 4.158.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads()` `[inline]`, `[inherited]`

Find out desired number of threads.

## Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

4.158.2.2 `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex __num_threads )` [inline],  
[inherited]

Set the desired number of threads.

## Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

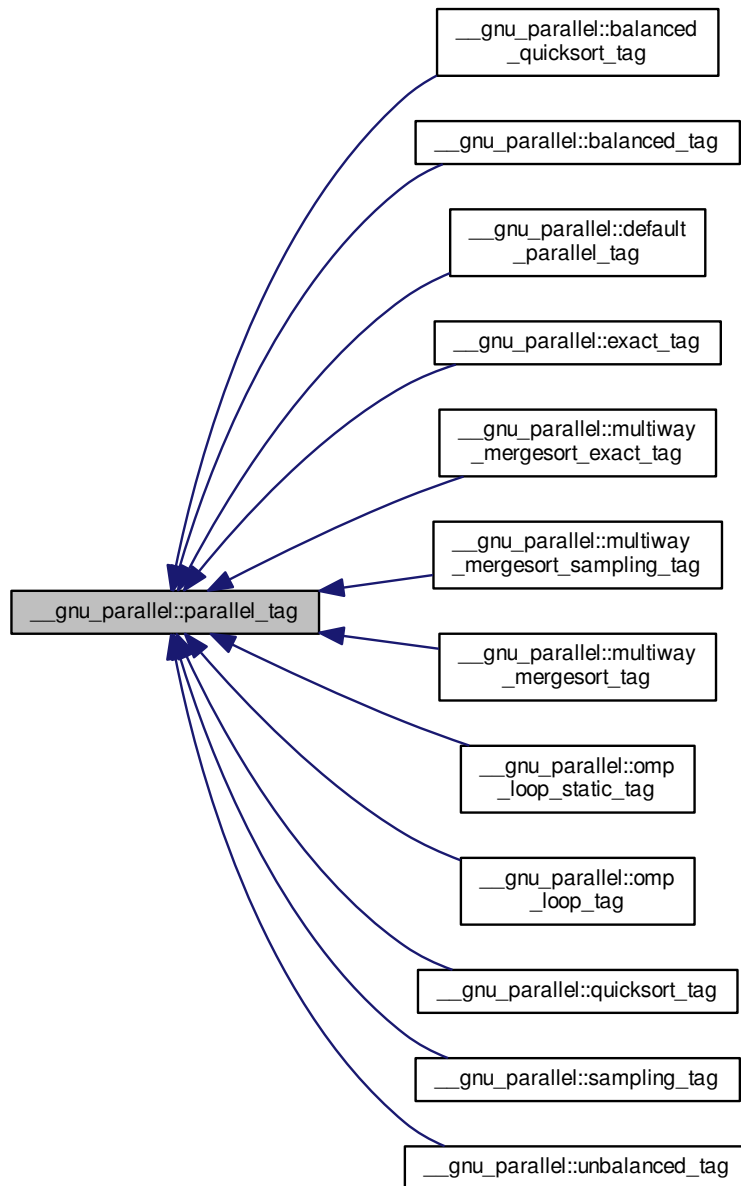
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 4.159 \_\_gnu\_parallel::parallel\_tag Struct Reference

Inheritance diagram for \_\_gnu\_parallel::parallel\_tag:



#### Public Member Functions

- [parallel\\_tag](#) ()
- [parallel\\_tag](#) ([\\_ThreadIndex](#) \_\_num\_threads)



- [\\_ThreadIndex \\_\\_get\\_num\\_threads \( \)](#)
- void [set\\_num\\_threads \( \\_ThreadIndex \\_\\_num\\_threads\)](#)

#### 4.159.1 Detailed Description

Recommends parallel execution at compile time, optionally using a user-specified number of threads.

Definition at line 46 of file tags.h.

#### 4.159.2 Constructor & Destructor Documentation

##### 4.159.2.1 [\\_\\_gnu\\_parallel::parallel\\_tag::parallel\\_tag \( \)](#) `[inline]`

Default constructor. Use default number of threads.

Definition at line 53 of file tags.h.

##### 4.159.2.2 [\\_\\_gnu\\_parallel::parallel\\_tag::parallel\\_tag \( \\_ThreadIndex \\_\\_num\\_threads \)](#) `[inline]`

Default constructor. Recommend number of threads to use.

Parameters

<a href="#">__num_threads</a>	Desired number of threads.
-------------------------------	----------------------------

Definition at line 58 of file tags.h.

#### 4.159.3 Member Function Documentation

##### 4.159.3.1 [\\_ThreadIndex \\_\\_gnu\\_parallel::parallel\\_tag::\\_\\_get\\_num\\_threads \( \)](#) `[inline]`

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#).

##### 4.159.3.2 [void \\_\\_gnu\\_parallel::parallel\\_tag::set\\_num\\_threads \( \\_ThreadIndex \\_\\_num\\_threads \)](#) `[inline]`

Set the desired number of threads.

Parameters

<a href="#">__num_threads</a>	Desired number of threads.
-------------------------------	----------------------------

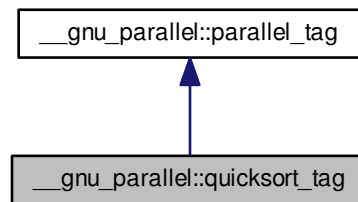
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.160 `__gnu_parallel::quicksort_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::quicksort_tag`:



## Public Member Functions

- **`quicksort_tag`** (`_ThreadIndex` `__num_threads`)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` `__num_threads`)

## 4.160.1 Detailed Description

Forces parallel sorting using unbalanced quicksort at compile time.

Definition at line 155 of file `tags.h`.

## 4.160.2 Member Function Documentation

4.160.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads` ( ) `[inline]`, `[inherited]`

Find out desired number of threads.

## Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::__parallel_sort`().

4.160.2.2 void `__gnu_parallel::parallel_tag::set_num_threads` ( `_ThreadIndex` `__num_threads` ) `[inline]`, `[inherited]`

Set the desired number of threads.

## Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

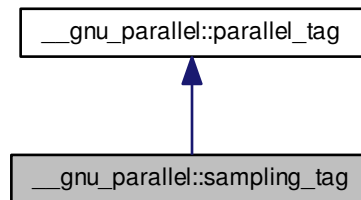
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

#### 4.161 `__gnu_parallel::sampling_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::sampling_tag`:



## Public Member Functions

- **sampling\_tag** (`_ThreadIndex` \_\_num\_threads)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` \_\_num\_threads)

##### 4.161.1 Detailed Description

Forces parallel merging with exact splitting, at compile time.

Definition at line 118 of file tags.h.

##### 4.161.2 Member Function Documentation

###### 4.161.2.1 `_ThreadIndex` `__gnu_parallel::parallel_tag::__get_num_threads` ( ) `[inline]`, `[inherited]`

Find out desired number of threads.

## Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort`().

4.161.2.2 `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex __num_threads )` [inline],  
[inherited]

Set the desired number of threads.

## Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 4.162 `__gnu_parallel::sequential_tag` Struct Reference

### 4.162.1 Detailed Description

Forces sequential execution at compile time.

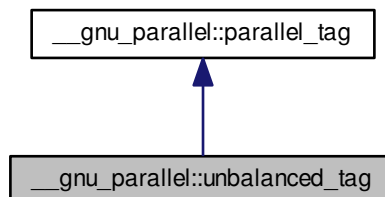
Definition at line 42 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 4.163 `__gnu_parallel::unbalanced_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::unbalanced_tag`:



## Public Member Functions

- [\\_ThreadIndex](#) `__get_num_threads` ()
- void `set_num_threads` ([\\_ThreadIndex](#) `__num_threads`)

### 4.163.1 Detailed Description

Recommends parallel execution using static load-balancing at compile time.

Definition at line 92 of file tags.h.

## 4.163.2 Member Function Documentation

#### 4.163.2.1 `__ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( )` `[inline]`, `[inherited]`

Find out desired number of threads.

## Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

```
4.163.2.2 void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex __num_threads ) [inline],
[inherited]
```

Set the desired number of threads.

### Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

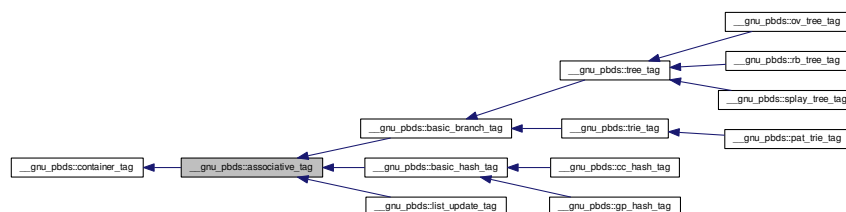
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- tags.h

#### 4.164 `__gnu_pbds::associative_tag` Struct Reference

Inheritance diagram for `gnu_pbds::associative_tag`:



#### 4.164.1 Detailed Description

Basic associative-container.

Definition at line 135 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- tag\_and\_trait.hpp

#### 4.165 `__gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >` Class Template Reference

Inherits type< Key, Mapped, \_Alloc, Tag, Policy\_Tl >.

##### Public Types

- typedef Node\_Update **node\_update**

##### Protected Member Functions

- **basic\_branch** (const [basic\\_branch](#) &other)
- template<typename T0 > **basic\_branch** (T0 t0)
- template<typename T0, typename T1 > **basic\_branch** (T0 t0, T1 t1)
- template<typename T0, typename T1, typename T2 > **basic\_branch** (T0 t0, T1 t1, T2 t2)
- template<typename T0, typename T1, typename T2, typename T3 > **basic\_branch** (T0 t0, T1 t1, T2 t2, T3 t3)
- template<typename T0, typename T1, typename T2, typename T3, typename T4 > **basic\_branch** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4)
- template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5 > **basic\_branch** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5)
- template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5, typename T6 > **basic\_branch** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6)

##### 4.165.1 Detailed Description

template<typename Key, typename Mapped, typename Tag, typename Node\_Update, typename Policy\_Tl, typename \_Alloc>class `__gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >`

A branched, tree-like (tree, trie) container abstraction.

##### Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Tag</i>	Instantiating data structure type, see container_tag.
<i>Node_Update</i>	Updates nodes, restores invariants.
<i>Policy_Tl</i>	Policy typelist.
<i>_Alloc</i>	Allocator type.

Base is dispatched at compile time via Tag, from the following choices: `tree_tag`, `trie_tag`, and their descendants.

Base choices are: `detail::ov_tree_map`, `detail::rb_tree_map`, `detail::splay_tree_map`, and `detail::pat_trie_map`.

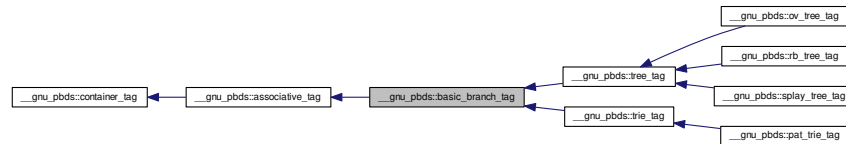
Definition at line 555 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 4.166 \_\_gnu\_pbds::basic\_branch\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::basic\_branch\_tag:



## 4.166.1 Detailed Description

Basic branch structure.

Definition at line 147 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.167 \_\_gnu\_pbds::basic\_hash\_table&lt; Key, Mapped, Hash\_Fn, Eq\_Fn, Resize\_Policy, Store\_Hash, Tag, Policy\_Tl, \_Alloc &gt; Class Template Reference

Inherits type< Key, Mapped, \_Alloc, Tag, \_\_gnu\_cxx::typelist::append< \_\_gnu\_cxx::typelist::create4< Hash\_Fn, Eq\_Fn, Resize\_Policy, detail::integral\_constant< int, Store\_Hash > >::type, Policy\_Tl >::type >.

## Protected Member Functions

- **basic\_hash\_table** (const [basic\\_hash\\_table](#) &other)
- template<typename T0 > **basic\_hash\_table** (T0 t0)
- template<typename T0 , typename T1 > **basic\_hash\_table** (T0 t0, T1 t1)
- template<typename T0 , typename T1 , typename T2 > **basic\_hash\_table** (T0 t0, T1 t1, T2 t2)
- template<typename T0 , typename T1 , typename T2 , typename T3 > **basic\_hash\_table** (T0 t0, T1 t1, T2 t2, T3 t3)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 > **basic\_hash\_table** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 > **basic\_hash\_table** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 > **basic\_hash\_table** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 > **basic\_hash\_table** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 , typename T8 > **basic\_hash\_table** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7, T8 t8)

## 4.167.1 Detailed Description



```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename Resize_Policy, bool Store_Hash, type-
name Tag, typename Policy_TI, typename _Alloc>class __gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_↵
Policy, Store_Hash, Tag, Policy_TI, _Alloc >
```

A hashed container abstraction.

## Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Hash_Fn</i>	Hashing functor.
<i>Eq_Fn</i>	Equal functor.
<i>Resize_Policy</i>	Resizes hash.
<i>Store_Hash</i>	Indicates whether the hash value will be stored along with each key.
<i>Tag</i>	Instantiating data structure type, see <code>container_tag</code> .
<i>Policy_TL</i>	Policy typelist.
<i>_Alloc</i>	Allocator type.

Base is dispatched at compile time via `Tag`, from the following choices: `cc_hash_tag`, `gp_hash_tag`, and descendants of `basic_hash_tag`.

Base choices are: `detail::cc_ht_map`, `detail::gp_ht_map`

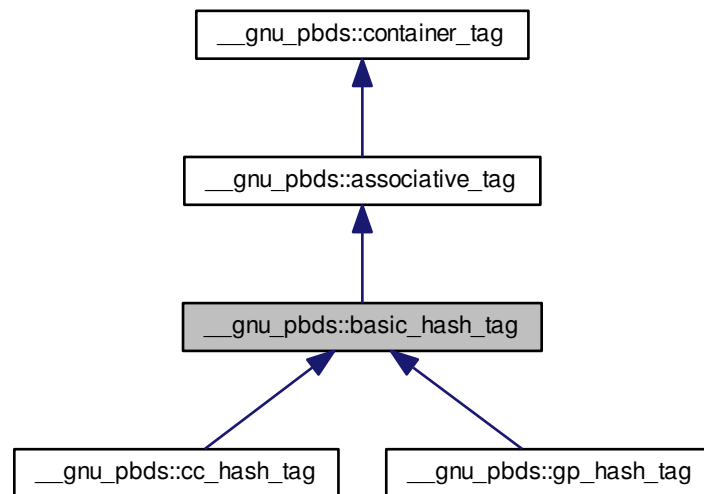
Definition at line 104 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

4.168 `__gnu_pbds::basic_hash_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::basic_hash_tag`:



## 4.168.1 Detailed Description

Basic hash structure.

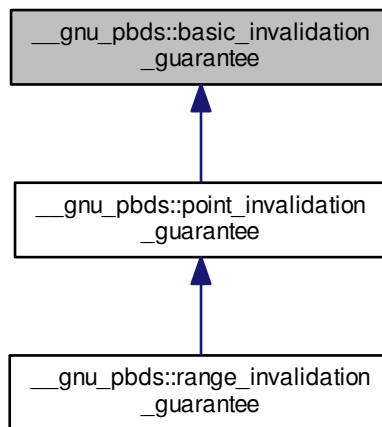
Definition at line 138 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.169 `__gnu_pbds::basic_invalidation_guarantee` Struct Reference

Inheritance diagram for `__gnu_pbds::basic_invalidation_guarantee`:



##### 4.169.1 Detailed Description

Signifies a basic invalidation guarantee that any iterator, pointer, or reference to a container object's mapped value type is valid as long as the container is not modified.

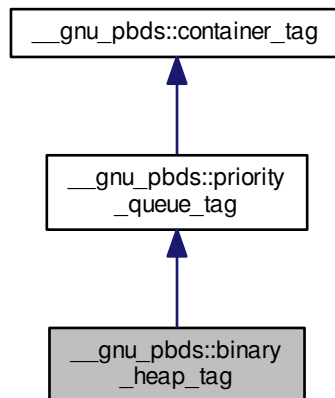
Definition at line 93 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

4.170 `__gnu_pbds::binary_heap_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::binary_heap_tag`:



## 4.170.1 Detailed Description

Binary-heap (array-based).

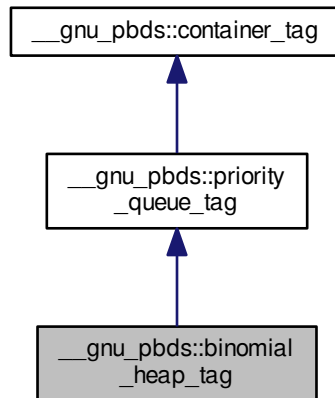
Definition at line 183 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.171 `__gnu_pbds::binomial_heap_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::binomial_heap_tag`:



##### 4.171.1 Detailed Description

Binomial-heap.

Definition at line 177 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.172 `__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >` Class Template Reference

##### Public Types

- enum { [external\\_load\\_access](#) }
- typedef `Size_Type` **size\_type**

##### Public Member Functions

- [cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger](#) (float load=0.5)
- float [get\\_load](#) () const
- void [set\\_load](#) (float load)
- void **swap** ([cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger](#)< `External_Load_Access`, `Size_Type` > &other)

## Protected Member Functions

- bool `is_grow_needed` (size\_type size, size\_type num\_entries) const
- bool `is_resize_needed` () const
- void `notify_cleared` ()
- void `notify_erase_search_collision` ()
- void `notify_erase_search_end` ()
- void `notify_erase_search_start` ()
- void `notify_erased` (size\_type num\_entries)
- void `notify_externally_resized` (size\_type new\_size)
- void `notify_find_search_collision` ()
- void `notify_find_search_end` ()
- void `notify_find_search_start` ()
- void `notify_insert_search_collision` ()
- void `notify_insert_search_end` ()
- void `notify_insert_search_start` ()
- void `notify_inserted` (size\_type num\_entries)
- void `notify_resized` (size\_type new\_size)

### 4.172.1 Detailed Description

template<bool External\_Load\_Access = false, typename Size\_Type = std::size\_t>class `__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >`

A resize trigger policy based on collision checks. It keeps the simulated load factor lower than some given load factor.

Definition at line 293 of file `hash_policy.hpp`.

### 4.172.2 Member Enumeration Documentation

4.172.2.1 template<bool External\_Load\_Access = false, typename Size\_Type = std::size\_t> anonymous enum

#### Enumerator

**`external_load_access`** Specifies whether the load factor can be accessed externally. The two options have different trade-offs in terms of flexibility, genericity, and encapsulation.

Definition at line 298 of file `hash_policy.hpp`.

### 4.172.3 Constructor & Destructor Documentation

4.172.3.1 template<bool External\_Load\_Access, typename Size\_Type > `__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::cc_hash_max_collision_check_resize_trigger ( float load = 0.5 )`

Default constructor, or constructor taking load, a `__load` factor which it will attempt to maintain.

Definition at line 44 of file `hash_policy.hpp`.

## 4.172.4 Member Function Documentation

4.172.4.1 `template<bool External_Load_Access, typename Size_Type > float __gnu_pbds::cc_hash_max↵  
_collision_check_resize_trigger< External_Load_Access, Size_Type >::get_load ( ) const  
[inline]`

Returns the current load.

Definition at line 190 of file hash\_policy.hpp.

4.172.4.2 `template<bool External_Load_Access, typename Size_Type > bool __gnu_pbds::cc_hash_max_collision↵  
_check_resize_trigger< External_Load_Access, Size_Type >::is_grow_needed ( size_type size, size_type  
num_entries ) const [inline],[protected]`

Queries whether a grow is needed. This method is called only if this object indicated is needed.

Definition at line 133 of file hash\_policy.hpp.

4.172.4.3 `template<bool External_Load_Access, typename Size_Type > bool __gnu_pbds::cc_hash_max_collision↵  
_check_resize_trigger< External_Load_Access, Size_Type >::is_resize_needed ( ) const [inline],  
[protected]`

Queries whether a resize is needed.

Definition at line 127 of file hash\_policy.hpp.

4.172.4.4 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_↵  
max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_cleared ( )  
[protected]`

Notifies the table was cleared.

Definition at line 121 of file hash\_policy.hpp.

4.172.4.5 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_↵  
check_resize_trigger< External_Load_Access, Size_Type >::notify_erase_search_collision ( ) [inline],  
[protected]`

Notifies a search encountered a collision.

Definition at line 97 of file hash\_policy.hpp.

4.172.4.6 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_↵  
check_resize_trigger< External_Load_Access, Size_Type >::notify_erase_search_end ( ) [inline],  
[protected]`

Notifies a search ended.

Definition at line 103 of file hash\_policy.hpp.

4.172.4.7 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_↵  
check_resize_trigger< External_Load_Access, Size_Type >::notify_erase_search_start ( ) [inline],  
[protected]`

Notifies an erase search started.

Definition at line 91 of file hash\_policy.hpp.

4.172.4.8 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_↵  
_check_resize_trigger< External_Load_Access, Size_Type >::notify_erased ( size_type num_entries )  
[inline], [protected]`

Notifies an element was erased.

Definition at line 115 of file hash\_policy.hpp.

4.172.4.9 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_↵  
_check_resize_trigger< External_Load_Access, Size_Type >::notify_externally_resized ( size_type new_size )  
[protected]`

Notifies the table was resized externally.

Definition at line 172 of file hash\_policy.hpp.

4.172.4.10 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_↵  
check_resize_trigger< External_Load_Access, Size_Type >::notify_find_search_collision ( ) [inline],  
[protected]`

Notifies a search encountered a collision.

Definition at line 61 of file hash\_policy.hpp.

4.172.4.11 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_↵  
check_resize_trigger< External_Load_Access, Size_Type >::notify_find_search_end ( ) [inline],  
[protected]`

Notifies a search ended.

Definition at line 67 of file hash\_policy.hpp.

4.172.4.12 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_↵  
check_resize_trigger< External_Load_Access, Size_Type >::notify_find_search_start ( ) [inline],  
[protected]`

Notifies a find search started.

Definition at line 55 of file hash\_policy.hpp.

4.172.4.13 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_↵  
check_resize_trigger< External_Load_Access, Size_Type >::notify_insert_search_collision ( ) [inline],  
[protected]`

Notifies a search encountered a collision.

Definition at line 79 of file hash\_policy.hpp.

4.172.4.14 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_↵  
_check_resize_trigger< External_Load_Access, Size_Type >::notify_insert_search_end ( ) [inline],  
[protected]`

Notifies a search ended.

Definition at line 85 of file hash\_policy.hpp.



4.172.4.15 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision↔  
_check_resize_trigger< External_Load_Access, Size_Type >::notify_insert_search_start ( ) [inline],  
[protected]`

Notifies an insert search started.

Definition at line 73 of file hash\_policy.hpp.

4.172.4.16 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision↔  
_check_resize_trigger< External_Load_Access, Size_Type >::notify_inserted ( size_type num_entries )  
[inline], [protected]`

Notifies an element was inserted.

Definition at line 109 of file hash\_policy.hpp.

4.172.4.17 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision↔  
_check_resize_trigger< External_Load_Access, Size_Type >::notify_resized ( size_type new_size )  
[protected]`

Notifies the table was resized as a result of this object's signifying that a resize is needed.

Definition at line 139 of file hash\_policy.hpp.

4.172.4.18 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max↔  
_collision_check_resize_trigger< External_Load_Access, Size_Type >::set_load ( float load  
)`

Sets the load; does not resize the container.

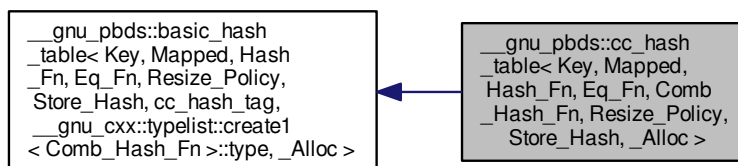
Definition at line 205 of file hash\_policy.hpp.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

#### 4.173 `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_↔ Hash, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >`:



## Public Types

- typedef Comb\_Hash\_Fn **comb\_hash\_fn**
- typedef [cc\\_hash\\_tag](#) **container\_category**
- typedef Eq\_Fn **eq\_fn**
- typedef Hash\_Fn **hash\_fn**
- typedef Resize\_Policy **resize\_policy**

## Public Member Functions

- [cc\\_hash\\_table](#) ()
- [cc\\_hash\\_table](#) (const hash\_fn &h)
- [cc\\_hash\\_table](#) (const hash\_fn &h, const eq\_fn &e)
- [cc\\_hash\\_table](#) (const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch)
- [cc\\_hash\\_table](#) (const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch, const resize\_policy &rp)
- template<typename It > [cc\\_hash\\_table](#) (It first, It last)
- template<typename It > [cc\\_hash\\_table](#) (It first, It last, const hash\_fn &h)
- template<typename It > [cc\\_hash\\_table](#) (It first, It last, const hash\_fn &h, const eq\_fn &e)
- template<typename It > [cc\\_hash\\_table](#) (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch)
- template<typename It > [cc\\_hash\\_table](#) (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch, const resize\_policy &rp)
- **cc\_hash\_table** (const [cc\\_hash\\_table](#) &other)
- [cc\\_hash\\_table](#) & **operator=** (const [cc\\_hash\\_table](#) &other)
- void **swap** ([cc\\_hash\\_table](#) &other)

## 4.173.1 Detailed Description

template<typename Key, typename Mapped, typename Hash\_Fn = typename detail::default\_hash\_fn<Key>::type, typename Eq\_Fn = typename detail::default\_eq\_fn<Key>::type, typename Comb\_Hash\_Fn = detail::default\_comb\_hash\_fn::type, typename Resize\_Policy = typename detail::default\_resize\_policy<Comb\_Hash\_Fn>::type, bool Store\_Hash = detail::default\_store\_hash, typename \_Alloc = std::allocator<char>> class `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >`

A collision-chaining hash-based associative container.

## Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Hash_Fn</i>	Hashing functor.
<i>Eq_Fn</i>	Equal functor.
<i>Comb_Hash_Fn</i>	Combining hash functor. If Hash_Fn is not null_type, then this is the ranged-hash functor; otherwise, this is the range-hashing functor. XXX(See Design::Hash-Based Containers::Hash Policies.)
<i>Resize_Policy</i>	Resizes hash.
<i>Store_Hash</i>	Indicates whether the hash value will be stored along with each key. If Hash_Fn is null_type, then the container will not compile if this value is true

<code>_Alloc</code>	Allocator type.
---------------------	-----------------

Base tag choices are: `cc_hash_tag`.

Base is `basic_hash_table`.

Definition at line 204 of file `assoc_container.hpp`.

#### 4.173.2 Constructor & Destructor Documentation

4.173.2.1 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table ( ) [inline]`

Default constructor.

Definition at line 217 of file `assoc_container.hpp`.

4.173.2.2 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table ( const hash_fn & h ) [inline]`

Constructor taking some policy objects. `r_hash_fn` will be copied by the `Hash_Fn` object of the container object.

Definition at line 221 of file `assoc_container.hpp`.

4.173.2.3 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table ( const hash_fn & h, const eq_fn & e ) [inline]`

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, and `r_eq_fn` will be copied by the `eq_fn` object of the container object.

Definition at line 228 of file `assoc_container.hpp`.

4.173.2.4 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table ( const hash_fn & h, const eq_fn & e, const comb_hash_fn & ch ) [inline]`

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, and `r_comb_hash_fn` will be copied by the `comb_hash_fn` object of the container object.

Definition at line 236 of file `assoc_container.hpp`.

4.173.2.5 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table ( const hash_fn & h, const eq_fn & e, const comb_hash_fn & ch, const resize_policy & rp ) [inline]`

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_hash_fn` will be copied by the `comb_hash_fn` object of the container object, and `r_resize_policy` will be copied by the `resize_policy` object of the container object.

Definition at line 245 of file `assoc_container.hpp`.

4.173.2.6 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table ( It first, It last ) [inline]`

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

Definition at line 253 of file `assoc_container.hpp`.

4.173.2.7 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table ( It first, It last, const hash_fn & h ) [inline]`

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

Definition at line 260 of file `assoc_container.hpp`.

4.173.2.8 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table ( It first, It last, const hash_fn & h, const eq_fn & e ) [inline]`

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, and `r_eq_fn` will be copied by the `eq_fn` object of the container object.

Definition at line 271 of file `assoc_container.hpp`.

```

4.173.2.9  template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
            typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_
            _hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool
            Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It >
            __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc
            >::cc_hash_table ( It first, It last, const hash_fn & h, const eq_fn & e, const comb_hash_fn & ch ) [inline]

```

Constructor taking \_\_iterators to a range of value\_types and some policy objects The value\_types between first\_it and last\_it will be inserted into the container object. r\_hash\_fn will be copied by the hash\_fn object of the container object, r\_eq\_fn will be copied by the eq\_fn object of the container object, and r\_comb\_hash\_fn will be copied by the comb\_hash\_fn object of the container object.

Definition at line 283 of file assoc\_container.hpp.

```

4.173.2.10 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
            typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_
            comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type,
            bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It >
            __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
            _Alloc >::cc_hash_table ( It first, It last, const hash_fn & h, const eq_fn & e, const comb_hash_fn & ch, const
            resize_policy & rp ) [inline]

```

Constructor taking \_\_iterators to a range of value\_types and some policy objects The value\_types between first\_it and last\_it will be inserted into the container object. r\_hash\_fn will be copied by the hash\_fn object of the container object, r\_eq\_fn will be copied by the eq\_fn object of the container object, r\_comb\_hash\_fn will be copied by the comb\_hash\_fn object of the container object, and r\_resize\_policy will be copied by the resize\_policy object of the container object.

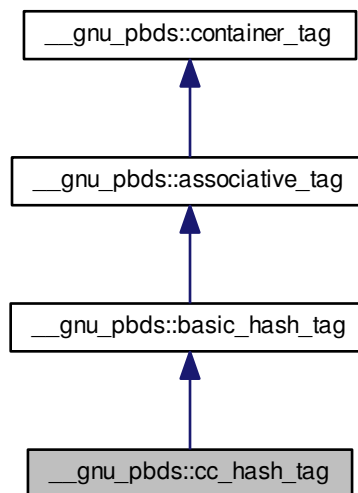
Definition at line 297 of file assoc\_container.hpp.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

4.174 `__gnu_pbds::cc_hash_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::cc_hash_tag`:



## 4.174.1 Detailed Description

Collision-chaining hash.

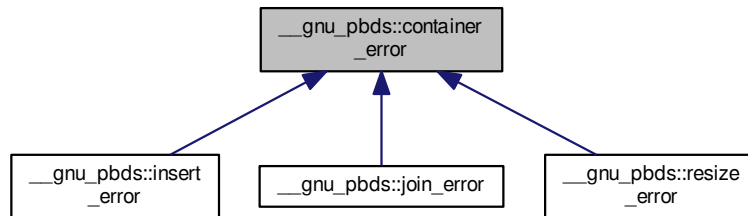
Definition at line 141 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.175 `__gnu_pbds::container_error` Struct Reference

Inheritance diagram for `__gnu_pbds::container_error`:



##### 4.175.1 Detailed Description

Base class for exceptions.

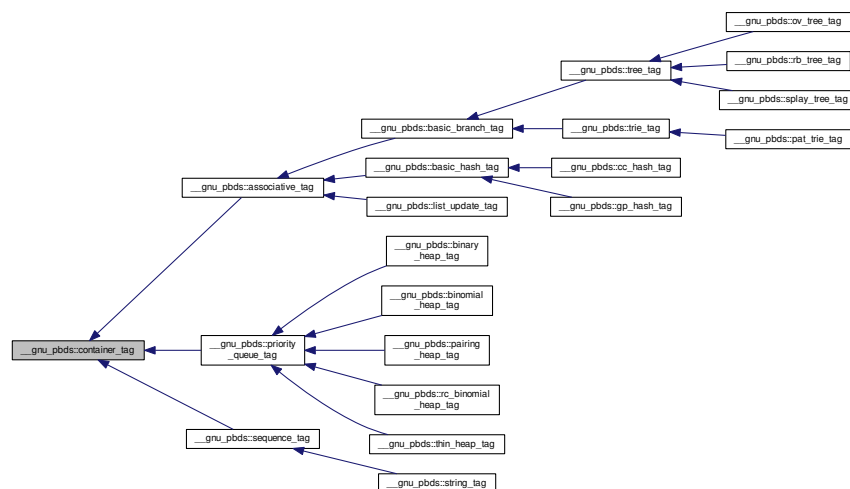
Definition at line 57 of file `exception.hpp`.

The documentation for this struct was generated from the following file:

- [exception.hpp](#)

#### 4.176 `__gnu_pbds::container_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::container_tag`:



## 4.176.1 Detailed Description

Base data structure tag.

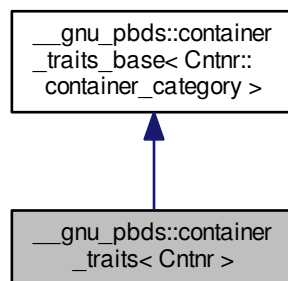
Definition at line 125 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

4.177 `__gnu_pbds::container_traits< Cntnr >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::container_traits< Cntnr >`:



## Public Types

- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- typedef `container_traits_base< container_category >` **base\_type**
- typedef `Cntnr::container_category` **container\_category**
- typedef `Cntnr` **container\_type**
- typedef `base_type::invalidation_guarantee` **invalidation\_guarantee**

## 4.177.1 Detailed Description

```
template<typename Cntnr>struct __gnu_pbds::container_traits< Cntnr >
```

Container traits.

Definition at line 418 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)



#### 4.178 `__gnu_pbds::container_traits_base<_Tag>` Struct Template Reference

##### 4.178.1 Detailed Description

```
template<typename _Tag> struct __gnu_pbds::container_traits_base<_Tag>
```

Primary template, container traits base.

Definition at line 220 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.179 `__gnu_pbds::container_traits_base<binary_heap_tag>` Struct Template Reference

##### Public Types

- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- typedef [binary\\_heap\\_tag](#) `container_category`
- typedef [basic\\_invalidation\\_guarantee](#) `invalidation_guarantee`

##### 4.179.1 Detailed Description

```
template<> struct __gnu_pbds::container_traits_base<binary_heap_tag>
```

Specialization, binary heap.

Definition at line 400 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.180 `__gnu_pbds::container_traits_base<binomial_heap_tag>` Struct Template Reference

##### Public Types

- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- typedef [binomial\\_heap\\_tag](#) `container_category`
- typedef [point\\_invalidation\\_guarantee](#) `invalidation_guarantee`

##### 4.180.1 Detailed Description

```
template<> struct __gnu_pbds::container_traits_base<binomial_heap_tag>
```

Specialization, binomial heap.

Definition at line 368 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.181 `__gnu_pbds::container_traits_base< cc_hash_tag >` Struct Template Reference

### Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef [cc\\_hash\\_tag](#) **container\_category**
- typedef [point\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

#### 4.181.1 Detailed Description

template<>struct `__gnu_pbds::container_traits_base< cc_hash_tag >`

Specialization, cc hash.

Definition at line 224 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.182 `__gnu_pbds::container_traits_base< gp_hash_tag >` Struct Template Reference

### Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef [gp\\_hash\\_tag](#) **container\_category**
- typedef [basic\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

#### 4.182.1 Detailed Description

template<>struct `__gnu_pbds::container_traits_base< gp_hash_tag >`

Specialization, gp hash.

Definition at line 240 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.183 `__gnu_pbds::container_traits_base< list_update_tag >` Struct Template Reference

### Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef [list\\_update\\_tag](#) **container\_category**
- typedef [point\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

#### 4.183.1 Detailed Description

`template<>struct __gnu_pbds::container_traits_base< list_update_tag >`

Specialization, list update.

Definition at line 320 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 4.184 `__gnu_pbds::container_traits_base< ov_tree_tag >` Struct Template Reference

#### Public Types

- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- typedef `ov_tree_tag` `container_category`
- typedef `basic_invalidation_guarantee` `invalidation_guarantee`

#### 4.184.1 Detailed Description

`template<>struct __gnu_pbds::container_traits_base< ov_tree_tag >`

Specialization, ov tree.

Definition at line 288 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 4.185 `__gnu_pbds::container_traits_base< pairing_heap_tag >` Struct Template Reference

#### Public Types

- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- typedef `pairing_heap_tag` `container_category`
- typedef `point_invalidation_guarantee` `invalidation_guarantee`

#### 4.185.1 Detailed Description

`template<>struct __gnu_pbds::container_traits_base< pairing_heap_tag >`

Specialization, pairing heap.

Definition at line 336 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.186 `__gnu_pbds::container_traits_base< pat_trie_tag >` Struct Template Reference

### Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef [pat\\_trie\\_tag](#) **container\_category**
- typedef [range\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

### 4.186.1 Detailed Description

`template<> struct __gnu_pbds::container_traits_base< pat_trie_tag >`

Specialization, pat trie.

Definition at line 304 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.187 `__gnu_pbds::container_traits_base< rb_tree_tag >` Struct Template Reference

### Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef [rb\\_tree\\_tag](#) **container\_category**
- typedef [range\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

### 4.187.1 Detailed Description

`template<> struct __gnu_pbds::container_traits_base< rb_tree_tag >`

Specialization, rb tree.

Definition at line 256 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.188 `__gnu_pbds::container_traits_base< rc_binomial_heap_tag >` Struct Template Reference

### Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef [rc\\_binomial\\_heap\\_tag](#) **container\_category**
- typedef [point\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

#### 4.188.1 Detailed Description

`template<>struct __gnu_pbds::container_traits_base< rc_binomial_heap_tag >`

Specialization, rc binomial heap.

Definition at line 384 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 4.189 `__gnu_pbds::container_traits_base< splay_tree_tag >` Struct Template Reference

#### Public Types

- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- typedef `splay_tree_tag` `container_category`
- typedef `range_invalidation_guarantee` `invalidation_guarantee`

#### 4.189.1 Detailed Description

`template<>struct __gnu_pbds::container_traits_base< splay_tree_tag >`

Specialization, splay tree.

Definition at line 272 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 4.190 `__gnu_pbds::container_traits_base< thin_heap_tag >` Struct Template Reference

#### Public Types

- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- typedef `thin_heap_tag` `container_category`
- typedef `point_invalidation_guarantee` `invalidation_guarantee`

#### 4.190.1 Detailed Description

`template<>struct __gnu_pbds::container_traits_base< thin_heap_tag >`

Specialization, thin heap.

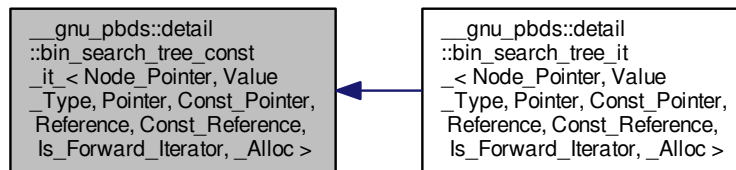
Definition at line 352 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.191 `__gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >`:



#### Public Types

- typedef Const\_Pointer **const\_pointer**
- typedef Const\_Reference **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `std::bidirectional_iterator_tag` **iterator\_category**
- typedef Pointer **pointer**
- typedef Reference **reference**
- typedef Value\_Type **value\_type**

#### Public Member Functions

- **bin\_search\_tree\_const\_it\_** (const Node\_Pointer p\_nd=0)
- **bin\_search\_tree\_const\_it\_** (const [bin\\_search\\_tree\\_const\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other)
- bool **operator!=** (const [bin\\_search\\_tree\\_const\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other) const
- bool **operator!=** (const [bin\\_search\\_tree\\_const\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other) const
- const\_reference **operator\*** () const
- [bin\\_search\\_tree\\_const\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > & **operator++** ()
- [bin\\_search\\_tree\\_const\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > **operator++** (int)
- [bin\\_search\\_tree\\_const\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > & **operator--** ()
- [bin\\_search\\_tree\\_const\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > **operator--** (int)
- const\_pointer **operator->** () const
- [bin\\_search\\_tree\\_const\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > & **operator=** (const [bin\\_search\\_tree\\_const\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other)

- [bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc > & operator=](#) (const [bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc > &other](#))
- bool **operator==** (const [bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc > &other](#)) const
- bool **operator==** (const [bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc > &other](#)) const

#### Public Attributes

- Node\_Pointer **m\_p\_nd**

#### Protected Member Functions

- void **dec** (false\_type)
- void **dec** (true\_type)
- void **inc** (false\_type)
- void **inc** (true\_type)

#### 4.191.1 Detailed Description

template<typename Node\_Pointer, typename Value\_Type, typename Pointer, typename Const\_Pointer, typename Reference, typename Const\_Reference, bool Is\_Forward\_Iterator, typename \_Alloc>class \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_it\_< Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc >

Const iterator.

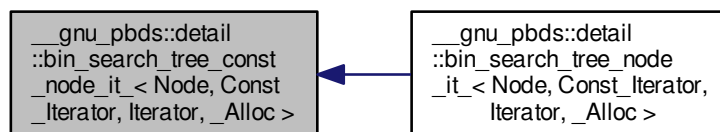
Definition at line 105 of file point\_iterators.hpp.

The documentation for this class was generated from the following file:

- [point\\_iterators.hpp](#)

#### 4.192 \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc > Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc >:



## Public Types

- typedef Const\_Iterator `const_reference`
- typedef `trivial_iterator_difference_type` `difference_type`
- typedef `trivial_iterator_tag` `iterator_category`
- typedef `_Alloc::template rebind< metadata_type >::other::const_reference` `metadata_const_reference`
- typedef Node::metadata\_type `metadata_type`
- typedef Const\_Iterator `reference`
- typedef Const\_Iterator `value_type`

## Public Member Functions

- `bin_search_tree_const_node_it_` (const node\_pointer p\_nd=0)
- `bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >` `get_l_child` () const
- `metadata_const_reference` `get_metadata` () const
- `bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >` `get_r_child` () const
- bool `operator!=` (const `bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >` &other) const
- `const_reference` `operator*` () const
- bool `operator==` (const `bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >` &other) const

## Public Attributes

- node\_pointer `m_p_nd`

### 4.192.1 Detailed Description

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>class __gnu_pbds::detail::bin_search_tree_const_↵
_node_it_< Node, Const_Iterator, Iterator, _Alloc >
```

Const node iterator.

Definition at line 58 of file `bin_search_tree_/node_iterators.hpp`.

### 4.192.2 Member Typedef Documentation

4.192.2.1 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Const_Iterator`  
`__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc`  
`>::const_reference`

Iterator's `__const` reference type.

Definition at line 80 of file `bin_search_tree_/node_iterators.hpp`.

4.192.2.2 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef`  
`trivial_iterator_difference_type __gnu_pbds::detail::bin_search_tree_const_node_it_< Node,`  
`Const_Iterator, Iterator, _Alloc >::difference_type`

Difference type.

Definition at line 71 of file `bin_search_tree_/node_iterators.hpp`.



4.192.2.3 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef trivial_iterator_tag  
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc  
>::iterator_category`

Category.

Definition at line 68 of file `bin_search_tree_/node_iterators.hpp`.

4.192.2.4 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef _Alloc::template  
rebind<metadata_type>::other::const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_<  
Node, Const_Iterator, Iterator, _Alloc >::metadata_const_reference`

Const metadata reference type.

Definition at line 88 of file `bin_search_tree_/node_iterators.hpp`.

4.192.2.5 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Node::metadata_type  
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc  
>::metadata_type`

Metadata type.

Definition at line 83 of file `bin_search_tree_/node_iterators.hpp`.

4.192.2.6 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Const_Iterator  
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::reference`

Iterator's reference type.

Definition at line 77 of file `bin_search_tree_/node_iterators.hpp`.

4.192.2.7 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Const_Iterator  
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::value_type`

Iterator's value type.

Definition at line 74 of file `bin_search_tree_/node_iterators.hpp`.

#### 4.192.3 Member Function Documentation

4.192.3.1 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bin_search_tree_const_node_<  
_it_<Node, Const_Iterator, Iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_const_node_it_< Node,  
Const_Iterator, Iterator, _Alloc >::get_l_child ( ) const [inline]`

Returns the `__const` node iterator associated with the left node.

Definition at line 107 of file `bin_search_tree_/node_iterators.hpp`.

4.192.3.2 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > metadata_const_reference  
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::get_metadata  
( ) const [inline]`

Metadata access.

Definition at line 102 of file `bin_search_tree_/node_iterators.hpp`.

4.192.3.3 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bin_search_tree_const_node_it<Node, Const_Iterator, Iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_const_node_it< Node, Const_Iterator, Iterator, _Alloc >::get_r_child ( ) const [inline]`

Returns the `__const` node iterator associated with the right node.

Definition at line 112 of file `bin_search_tree_/node_iterators.hpp`.

4.192.3.4 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bool __gnu_pbds::detail::bin_search_tree_const_node_it< Node, Const_Iterator, Iterator, _Alloc >::operator!=( const bin_search_tree_const_node_it< Node, Const_Iterator, Iterator, _Alloc > & other ) const [inline]`

Compares (negatively) to a different iterator object.

Definition at line 122 of file `bin_search_tree_/node_iterators.hpp`.

4.192.3.5 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > const_reference __gnu_pbds::detail::bin_search_tree_const_node_it< Node, Const_Iterator, Iterator, _Alloc >::operator*( ) const [inline]`

Access.

Definition at line 97 of file `bin_search_tree_/node_iterators.hpp`.

4.192.3.6 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bool __gnu_pbds::detail::bin_search_tree_const_node_it< Node, Const_Iterator, Iterator, _Alloc >::operator==( const bin_search_tree_const_node_it< Node, Const_Iterator, Iterator, _Alloc > & other ) const [inline]`

Compares to a different iterator object.

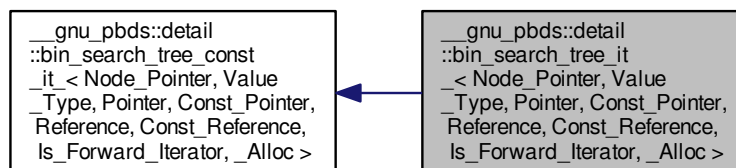
Definition at line 117 of file `bin_search_tree_/node_iterators.hpp`.

The documentation for this class was generated from the following file:

- [bin\\_search\\_tree\\_/node\\_iterators.hpp](#)

## 4.193 `__gnu_pbds::detail::bin_search_tree_it < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::bin_search_tree_it < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >`:



## Public Types

- typedef Const\_Pointer **const\_pointer**
- typedef Const\_Reference **const\_reference**
- typedef \_Alloc::difference\_type **difference\_type**
- typedef std::bidirectional\_iterator\_tag **iterator\_category**
- typedef Pointer **pointer**
- typedef Reference **reference**
- typedef Value\_Type **value\_type**

## Public Member Functions

- **bin\_search\_tree\_it\_** (const Node\_Pointer p\_nd=0)
- **bin\_search\_tree\_it\_** (const [bin\\_search\\_tree\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, !Is\_Forward\_Iterator, \_Alloc > &other)
- bool **operator!=** (const [bin\\_search\\_tree\\_const\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other) const
- bool **operator!=** (const [bin\\_search\\_tree\\_const\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, !Is\_Forward\_Iterator, \_Alloc > &other) const
- [bin\\_search\\_tree\\_const\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc >::reference **operator\*** () const
- [bin\\_search\\_tree\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_↔\_Forward\_Iterator, \_Alloc > & **operator++** ()
- [bin\\_search\\_tree\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_↔\_Forward\_Iterator, \_Alloc > **operator++** (int)
- [bin\\_search\\_tree\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_↔\_Forward\_Iterator, \_Alloc > & **operator--** ()
- [bin\\_search\\_tree\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_↔\_Forward\_Iterator, \_Alloc > **operator--** (int)
- [bin\\_search\\_tree\\_const\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc >::pointer **operator->** () const
- [bin\\_search\\_tree\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_↔\_Forward\_Iterator, \_Alloc > & **operator=** (const [bin\\_search\\_tree\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_↔\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other)
- [bin\\_search\\_tree\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_↔\_Forward\_Iterator, \_Alloc > & **operator=** (const [bin\\_search\\_tree\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_↔\_Pointer, Reference, Const\_Reference, !Is\_Forward\_Iterator, \_Alloc > &other)
- bool **operator==** (const [bin\\_search\\_tree\\_const\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other) const
- bool **operator==** (const [bin\\_search\\_tree\\_const\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, !Is\_Forward\_Iterator, \_Alloc > &other) const

## Public Attributes

- Node\_Pointer **m\_p\_nd**

## Protected Types

- typedef [bin\\_search\\_tree\\_const\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_↔\_Reference, Is\_Forward\_Iterator, \_Alloc > **base\_it\_type**

### Protected Member Functions

- void **dec** (false\_type)
- void **dec** (true\_type)
- void **inc** (false\_type)
- void **inc** (true\_type)

#### 4.193.1 Detailed Description

```
template<typename Node_Pointer, typename Value_Type, typename Pointer, typename Const_Pointer, typename Reference, typename Const_Reference, bool Is_Forward_Iterator, typename _Alloc> class __gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >
```

Iterator.

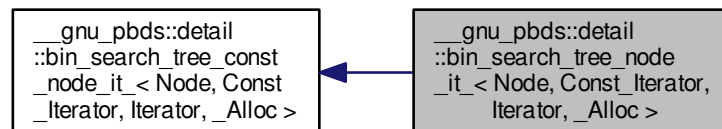
Definition at line 282 of file `point_iterators.hpp`.

The documentation for this class was generated from the following file:

- [point\\_iterators.hpp](#)

#### 4.194 `__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >`:



### Public Types

- typedef Iterator [const\\_reference](#)
- typedef [trivial\\_iterator\\_difference\\_type](#) difference\_type
- typedef [trivial\\_iterator\\_tag](#) iterator\_category
- typedef `_Alloc::template rebind< metadata\_type >::other::const_reference` [metadata\\_const\\_reference](#)
- typedef `Node::metadata_type` [metadata\\_type](#)
- typedef Iterator [reference](#)
- typedef Iterator [value\\_type](#)

## Public Member Functions

- **bin\_search\_tree\_node\_it\_** (const node\_pointer p\_nd=0)
- **bin\_search\_tree\_node\_it\_** < Node, Const\_iterator, Iterator, \_Alloc > **get\_l\_child** () const
- **metadata\_const\_reference** **get\_metadata** () const
- **bin\_search\_tree\_node\_it\_** < Node, Const\_iterator, Iterator, \_Alloc > **get\_r\_child** () const
- bool **operator!=** (const **bin\_search\_tree\_const\_node\_it\_** < Node, Const\_iterator, Iterator, \_Alloc > &other) const
- Iterator **operator\*** () const
- bool **operator==** (const **bin\_search\_tree\_const\_node\_it\_** < Node, Const\_iterator, Iterator, \_Alloc > &other) const

## Public Attributes

- node\_pointer **m\_p\_nd**

## 4.194.1 Detailed Description

```
template<typename Node, class Const_iterator, class Iterator, typename _Alloc>class __gnu_pbds::detail::bin_search_tree_node_it_ < Node, Const_iterator, Iterator, _Alloc >
```

Node iterator.

Definition at line 136 of file bin\_search\_tree\_/node\_iterators.hpp.

## 4.194.2 Member Typedef Documentation

```
4.194.2.1 template<typename Node , class Const_iterator , class Iterator , typename _Alloc > typedef Iterator
__gnu_pbds::detail::bin_search_tree_node_it_ < Node, Const_iterator, Iterator, _Alloc >::const_reference
```

Iterator's \_\_const reference type.

Definition at line 153 of file bin\_search\_tree\_/node\_iterators.hpp.

```
4.194.2.2 template<typename Node , class Const_iterator , class Iterator , typename _Alloc > typedef
trivial_iterator_difference_type __gnu_pbds::detail::bin_search_tree_const_node_it_ < Node,
Const_iterator, Iterator, _Alloc >::difference_type [inherited]
```

Difference type.

Definition at line 71 of file bin\_search\_tree\_/node\_iterators.hpp.

```
4.194.2.3 template<typename Node , class Const_iterator , class Iterator , typename _Alloc > typedef trivial_iterator_tag
__gnu_pbds::detail::bin_search_tree_const_node_it_ < Node, Const_iterator, Iterator, _Alloc
>::iterator_category [inherited]
```

Category.

Definition at line 68 of file bin\_search\_tree\_/node\_iterators.hpp.

```
4.194.2.4 template<typename Node , class Const_iterator , class Iterator , typename _Alloc > typedef _Alloc::template
rebind<metadata_type>::other::const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_ <
Node, Const_iterator, Iterator, _Alloc >::metadata_const_reference [inherited]
```

Const metadata reference type.

Definition at line 88 of file bin\_search\_tree\_/node\_iterators.hpp.

4.194.2.5 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Node::metadata_type  
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc  
>::metadata_type [inherited]`

Metadata type.

Definition at line 83 of file `bin_search_tree_/node_iterators.hpp`.

4.194.2.6 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Iterator  
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >::reference`

Iterator's reference type.

Definition at line 150 of file `bin_search_tree_/node_iterators.hpp`.

4.194.2.7 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Iterator  
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >::value_type`

Iterator's value type.

Definition at line 147 of file `bin_search_tree_/node_iterators.hpp`.

#### 4.194.3 Member Function Documentation

4.194.3.1 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bin_search_tree_node_it_↵  
<Node, Const_Iterator, Iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator,  
Iterator, _Alloc >::get_l_child ( ) const [inline]`

Returns the node iterator associated with the left node.

Definition at line 167 of file `bin_search_tree_/node_iterators.hpp`.

4.194.3.2 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > metadata_const_reference  
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::get_metadata  
( ) const [inline], [inherited]`

Metadata access.

Definition at line 102 of file `bin_search_tree_/node_iterators.hpp`.

4.194.3.3 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bin_search_tree_node_it_↵  
<Node, Const_Iterator, Iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator,  
Iterator, _Alloc >::get_r_child ( ) const [inline]`

Returns the node iterator associated with the right node.

Definition at line 175 of file `bin_search_tree_/node_iterators.hpp`.

4.194.3.4 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bool  
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::operator!= (   
const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > & other ) const [inline],  
[inherited]`

Compares (negatively) to a different iterator object.

Definition at line 122 of file `bin_search_tree_/node_iterators.hpp`.

4.194.3.5 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > Iterator  
 __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >::operator* ( ) const  
 [inline]`

Access.

Definition at line 162 of file `bin_search_tree_/node_iterators.hpp`.

4.194.3.6 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bool  
 __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::operator== (   
 const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > & other ) const [inline],  
 [inherited]`

Compares to a different iterator object.

Definition at line 117 of file `bin_search_tree_/node_iterators.hpp`.

The documentation for this class was generated from the following file:

- [bin\\_search\\_tree\\_/node\\_iterators.hpp](#)

#### 4.195 `__gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >` Struct Template Reference

##### Public Types

- typedef `bin_search_tree_const_it_< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **const\_reverse\_iterator**
- typedef `Node` **node**
- typedef `bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc >` **node\_const\_iterator**
- typedef `bin_search_tree_node_it_< Node, point_const_iterator, point_iterator, _Alloc >` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc >` **node\_update**
- typedef `__gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * null_node_update_pointer`
- typedef `bin_search_tree_const_it_< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point\_const\_iterator**
- typedef `bin_search_tree_it_< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point\_iterator**
- typedef `bin_search_tree_it_< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **reverse\_iterator**

##### 4.195.1 Detailed Description

`template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class Cmp_Fn, typename _Alloc > class Node_Update, class Node, typename _Alloc> struct __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >`

Binary search tree traits, primary template.

Definition at line 63 of file `bin_search_tree_/traits.hpp`.

## 4.195.2 Member Typedef Documentation

4.195.2.1 `template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class Cmp_Fn, typename _Alloc > class Node_Update, class Node, typename _Alloc> typedef bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >::node_const_iterator`

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 131 of file `bin_search_tree_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [bin\\_search\\_tree\\_/traits.hpp](#)

## 4.196 \_\_gnu\_pbds::detail::bin\_search\_tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, Node, \_Alloc > Struct Template Reference

## Public Types

- typedef `bin_search_tree_const_it_< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **const\_reverse\_iterator**
- typedef `Node` **node**
- typedef `bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc >` **node\_const\_iterator**
- typedef `node_const_iterator` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc >` **node\_update**
- typedef `__gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * null_node_update_pointer`
- typedef `bin_search_tree_const_it_< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point\_const\_iterator**
- typedef `point_const_iterator` **point\_iterator**
- typedef `const_reverse_iterator` **reverse\_iterator**

## 4.196.1 Detailed Description

`template<typename Key, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class Cmp_Fn, typename _Alloc > class Node_Update, class Node, typename _Alloc> struct __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >`

Specialization.

Definition at line 169 of file `bin_search_tree_/traits.hpp`.

## 4.196.2 Member Typedef Documentation

4.196.2.1 `template<typename Key, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class Cmp_Fn, typename _Alloc > class Node_Update, class Node, typename _Alloc > typedef bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >::node_const_iterator`

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.



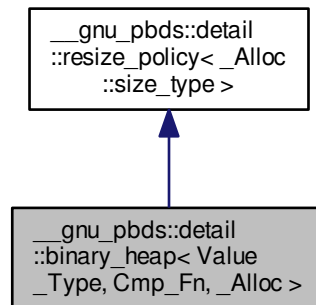
Definition at line 221 of file `bin_search_tree_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [bin\\_search\\_tree\\_/traits.hpp](#)

#### 4.197 `__gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >`:



#### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `cond_dealtor< value_type, _Alloc >` **cond\_dealtor\_t**
- typedef `binary_heap_const_iterator< value_type, entry, simple_value, _Alloc >` **const\_iterator**
- typedef `value_allocator::const_pointer` **const\_pointer**
- typedef `value_allocator::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `__conditional_type< simple_value, value_type, pointer >::__type` **entry**
- typedef `_Alloc::template rebind< entry >::other` **entry\_allocator**
- typedef `entry_cmp< Value_Type, Cmp_Fn, _Alloc, is_simple< Value_Type >::value >::type` **entry\_cmp**
- typedef `entry_allocator::pointer` **entry\_pointer**
- typedef `const_iterator` **iterator**
- typedef `binary_heap_point_const_iterator< value_type, entry, simple_value, _Alloc >` **point\_const\_iterator**
- typedef `point_const_iterator` **point\_iterator**
- typedef `value_allocator::pointer` **pointer**
- typedef `value_allocator::reference` **reference**
- typedef `__gnu_pbds::detail::resize_policy< typename _Alloc::size_type >` **resize\_policy**
- typedef `_Alloc::size_type` **size\_type**
- typedef `Value_Type` **value\_type**

## Public Member Functions

- **binary\_heap** (const cmp\_fn &)
- **binary\_heap** (const [binary\\_heap](#) &)
- [iterator](#) **begin** ()
- [const\\_iterator](#) **begin** () const
- void **clear** ()
- bool **empty** () const
- [iterator](#) **end** ()
- [const\\_iterator](#) **end** () const
- void **erase** ([point\\_iterator](#))
- void **erase\_at** (entry\_pointer, size\_type, false\_type)
- void **erase\_at** (entry\_pointer, size\_type, true\_type)
- template<typename Pred > size\_type **erase\_if** (Pred)
- Cmp\_Fn & **get\_cmp\_fn** ()
- const Cmp\_Fn & **get\_cmp\_fn** () const
- size\_type **get\_new\_size\_for\_arbitrary** (size\_type) const
- size\_type **get\_new\_size\_for\_grow** () const
- size\_type **get\_new\_size\_for\_shrink** () const
- bool **grow\_needed** (size\_type) const
- void **join** ([binary\\_heap](#) &)
- size\_type **max\_size** () const
- void **modify** ([point\\_iterator](#), const\_reference)
- void **notify\_arbitrary** (size\_type)
- void **notify\_grow\_resize** ()
- void **notify\_shrink\_resize** ()
- void **pop** ()
- [point\\_iterator](#) **push** (const\_reference)
- bool **resize\_needed\_for\_grow** (size\_type) const
- bool **resize\_needed\_for\_shrink** (size\_type) const
- bool **shrink\_needed** (size\_type) const
- size\_type **size** () const
- template<typename Pred > void **split** (Pred, [binary\\_heap](#) &)
- void **swap** ([resize\\_policy](#)< \_Alloc::size\_type > &)
- void **swap** ([binary\\_heap](#) &)
- const\_reference **top** () const

## Static Public Attributes

- static const \_Alloc::size\_type **min\_size**

## Protected Member Functions

- template<typename It > void **copy\_from\_range** (It, It)

#### 4.197.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>class __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >
```

Binary heaps composed of resize and compare policies.

Based on CLRS.

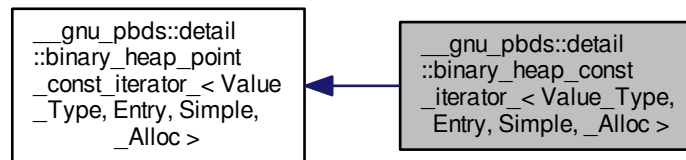
Definition at line 84 of file binary\_heap.hpp.

The documentation for this class was generated from the following file:

- [binary\\_heap.hpp](#)

#### 4.198 \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc > Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc >:



#### Public Types

- typedef [base\\_type::const\\_pointer](#) `const_pointer`
- typedef [base\\_type::const\\_reference](#) `const_reference`
- typedef [\\_Alloc::difference\\_type](#) `difference_type`
- typedef [std::forward\\_iterator\\_tag](#) `iterator_category`
- typedef [base\\_type::pointer](#) `pointer`
- typedef [base\\_type::reference](#) `reference`
- typedef [base\\_type::value\\_type](#) `value_type`

#### Public Member Functions

- **binary\_heap\_const\_iterator\_** (entry\_pointer p\_e)
- [binary\\_heap\\_const\\_iterator\\_](#) ()
- [binary\\_heap\\_const\\_iterator\\_](#) (const [binary\\_heap\\_const\\_iterator\\_](#) &other)
- bool [operator!=](#) (const [binary\\_heap\\_const\\_iterator\\_](#) &other) const
- bool [operator!=](#) (const [binary\\_heap\\_point\\_const\\_iterator\\_](#) &other) const
- [const\\_reference](#) [operator\\*](#) () const
- [binary\\_heap\\_const\\_iterator\\_](#) & [operator++](#) ()

- `binary_heap_const_iterator_ operator++` (int)
- `const_pointer operator->` () const
- `bool operator==` (const `binary_heap_const_iterator_` &other) const
- `bool operator==` (const `binary_heap_point_const_iterator_` &other) const

#### Public Attributes

- entry\_pointer `m_p_e`

#### 4.198.1 Detailed Description

`template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>class __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >`

Const point-type iterator.

Definition at line 60 of file `binary_heap_/const_iterator.hpp`.

#### 4.198.2 Member Typedef Documentation

4.198.2.1 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef base_type::const_pointer __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::const_pointer`

Iterator's const pointer type.

Definition at line 80 of file `binary_heap_/const_iterator.hpp`.

4.198.2.2 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef base_type::const_reference __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::const_reference`

Iterator's const reference type.

Definition at line 86 of file `binary_heap_/const_iterator.hpp`.

4.198.2.3 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef _Alloc::difference_type __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::difference_type`

Difference type.

Definition at line 71 of file `binary_heap_/const_iterator.hpp`.

4.198.2.4 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef std::forward_iterator_tag __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::iterator_category`

Category.

Definition at line 68 of file `binary_heap_/const_iterator.hpp`.

4.198.2.5 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef base_type::pointer __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::pointer`

Iterator's pointer type.

Definition at line 77 of file `binary_heap_/const_iterator.hpp`.

4.198.2.6 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef base_type::reference  
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::reference`

Iterator's reference type.

Definition at line 83 of file `binary_heap_/const_iterator.hpp`.

4.198.2.7 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef base_type::value_type  
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::value_type`

Iterator's value type.

Definition at line 74 of file `binary_heap_/const_iterator.hpp`.

#### 4.198.3 Constructor & Destructor Documentation

4.198.3.1 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > __gnu_pbds::detail::  
::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::binary_heap_const_iterator_ ( )  
[inline]`

Default constructor.

Definition at line 94 of file `binary_heap_/const_iterator.hpp`.

4.198.3.2 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > __gnu_pbds::detail::binary_  
_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::binary_heap_const_iterator_ ( const  
binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other ) [inline]`

Copy constructor.

Definition at line 99 of file `binary_heap_/const_iterator.hpp`.

#### 4.198.4 Member Function Documentation

4.198.4.1 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > bool  
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator!=( const  
binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other ) const [inline]`

Compares content (negatively) to a different iterator object.

Definition at line 110 of file `binary_heap_/const_iterator.hpp`.

4.198.4.2 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > bool  
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator!=(  
const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other ) const [inline],  
[inherited]`

Compares content (negatively) to a different iterator object.

Definition at line 126 of file `binary_heap_/point_const_iterator.hpp`.

4.198.4.3 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > const_reference  
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator* ( )  
const [inline], [inherited]`

Access.

Definition at line 113 of file `binary_heap_/point_const_iterator.hpp`.

4.198.4.4 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > const_pointer  
 __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator-> ( ) const [inline],[inherited]`

Access.

Definition at line 105 of file `binary_heap_/point_const_iterator.hpp`.

4.198.4.5 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > bool  
 __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator== ( const  
 binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other ) const [inline]`

Compares content to a different iterator object.

Definition at line 105 of file `binary_heap_/const_iterator.hpp`.

4.198.4.6 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > bool  
 __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator== ( const  
 binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other ) const [inline],  
 [inherited]`

Compares content to a different iterator object.

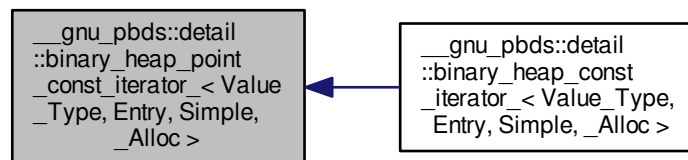
Definition at line 121 of file `binary_heap_/point_const_iterator.hpp`.

The documentation for this class was generated from the following file:

- [binary\\_heap\\_/const\\_iterator.hpp](#)

## 4.199 `__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >`:  
 >:



### Public Types

- `typedef _Alloc::template rebind< value\_type >::other::const_pointer const\_pointer`
- `typedef _Alloc::template rebind< value\_type >::other::const_reference const\_reference`
- `typedef trivial\_iterator\_difference\_type difference\_type`
- `typedef trivial\_iterator\_tag iterator\_category`

- typedef \_Alloc::template rebind< [value\\_type](#) >::other::pointer [pointer](#)
- typedef \_Alloc::template rebind< [value\\_type](#) >::other::reference [reference](#)
- typedef Value\_Type [value\\_type](#)

#### Public Member Functions

- **binary\_heap\_point\_const\_iterator\_** (entry\_pointer p\_e)
- [binary\\_heap\\_point\\_const\\_iterator\\_](#) ()
- [binary\\_heap\\_point\\_const\\_iterator\\_](#) (const [binary\\_heap\\_point\\_const\\_iterator\\_](#) &other)
- bool [operator!=](#) (const [binary\\_heap\\_point\\_const\\_iterator\\_](#) &other) const
- [const\\_reference operator\\*](#) () const
- [const\\_pointer operator->](#) () const
- bool [operator==](#) (const [binary\\_heap\\_point\\_const\\_iterator\\_](#) &other) const

#### Public Attributes

- entry\_pointer **m\_p\_e**

#### Protected Types

- typedef \_Alloc::template rebind< Entry >::other::pointer **entry\_pointer**

#### 4.199.1 Detailed Description

template<typename Value\_Type, typename Entry, bool Simple, typename \_Alloc>class \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc >

Const point-type iterator.

Definition at line 55 of file binary\_heap\_/point\_const\_iterator.hpp.

#### 4.199.2 Member Typedef Documentation

4.199.2.1 template<typename Value\_Type , typename Entry , bool Simple, typename \_Alloc > typedef \_Alloc::template rebind<[value\\_type](#)>::other::const\_pointer \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc >::const\_pointer

Iterator's const pointer type.

Definition at line 77 of file binary\_heap\_/point\_const\_iterator.hpp.

4.199.2.2 template<typename Value\_Type , typename Entry , bool Simple, typename \_Alloc > typedef \_Alloc::template rebind<[value\\_type](#)>::other::const\_reference \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc >::const\_reference

Iterator's const reference type.

Definition at line 87 of file binary\_heap\_/point\_const\_iterator.hpp.

4.199.2.3 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef  
trivial_iterator_difference_type __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type,  
Entry, Simple, _Alloc >::difference_type`

Difference type.

Definition at line 65 of file `binary_heap_/point_const_iterator.hpp`.

4.199.2.4 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef trivial_iterator_tag  
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc  
>::iterator_category`

Category.

Definition at line 62 of file `binary_heap_/point_const_iterator.hpp`.

4.199.2.5 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef _Alloc::template  
rebind<value_type>::other::pointer __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type,  
Entry, Simple, _Alloc >::pointer`

Iterator's pointer type.

Definition at line 72 of file `binary_heap_/point_const_iterator.hpp`.

4.199.2.6 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef _Alloc::template  
rebind<value_type>::other::reference __gnu_pbds::detail::binary_heap_point_const_iterator_<  
Value_Type, Entry, Simple, _Alloc >::reference`

Iterator's reference type.

Definition at line 82 of file `binary_heap_/point_const_iterator.hpp`.

4.199.2.7 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef Value_Type  
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::value_type`

Iterator's value type.

Definition at line 68 of file `binary_heap_/point_const_iterator.hpp`.

#### 4.199.3 Constructor & Destructor Documentation

4.199.3.1 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > __gnu_pbds::detail::binary_↵  
heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::binary_heap_point_const_iterator_( )  
[inline]`

Default constructor.

Definition at line 95 of file `binary_heap_/point_const_iterator.hpp`.

4.199.3.2 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > __gnu_pbds::detail::binary_↵  
heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::binary_heap_point_const_iterator_(  
const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other ) [inline]`

Copy constructor.

Definition at line 99 of file `binary_heap_/point_const_iterator.hpp`.



#### 4.199.4 Member Function Documentation

4.199.4.1 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > bool  
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator!=(  
const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other ) const [inline]`

Compares content (negatively) to a different iterator object.

Definition at line 126 of file `binary_heap_/point_const_iterator.hpp`.

4.199.4.2 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > const_reference  
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator*( )  
const [inline]`

Access.

Definition at line 113 of file `binary_heap_/point_const_iterator.hpp`.

4.199.4.3 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > const_pointer  
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator->(  
) const [inline]`

Access.

Definition at line 105 of file `binary_heap_/point_const_iterator.hpp`.

4.199.4.4 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > bool  
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator==(  
const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other ) const [inline]`

Compares content to a different iterator object.

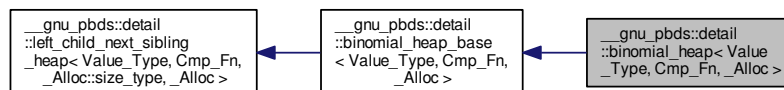
Definition at line 121 of file `binary_heap_/point_const_iterator.hpp`.

The documentation for this class was generated from the following file:

- [binary\\_heap\\_/point\\_const\\_iterator.hpp](#)

#### 4.200 \_\_gnu\_pbds::detail::binomial\_heap< Value\_Type, Cmp\_Fn, \_Alloc > Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >`:



#### Public Types

- `typedef base_type::allocator_type allocator_type`
- `typedef base_type::cmp_fn cmp_fn`
- `typedef base\_type::const\_iterator const_iterator`

- typedef `base_type::const_pointer` **const\_pointer**
- typedef `base_type::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::point_const_iterator` **point\_const\_iterator**
- typedef `base_type::point_iterator` **point\_iterator**
- typedef `base_type::pointer` **pointer**
- typedef `base_type::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `Value_Type` **value\_type**

#### Public Member Functions

- **binomial\_heap** (const Cmp\_Fn &)
- **binomial\_heap** (const `binomial_heap` &)
- `iterator` **begin** ()
- `const_iterator` **begin** () const
- void **clear** ()
- bool **empty** () const
- `iterator` **end** ()
- `const_iterator` **end** () const
- void **erase** (`point_iterator`)
- template<typename Pred > `size_type` **erase\_if** (Pred)
- Cmp\_Fn & **get\_cmp\_fn** ()
- const Cmp\_Fn & **get\_cmp\_fn** () const
- void **join** (`binomial_heap_base`< Value\_Type, Cmp\_Fn, \_Alloc > &)
- `size_type` **max\_size** () const
- void **modify** (`point_iterator`, const\_reference)
- void **pop** ()
- `point_iterator` **push** (const\_reference)
- `size_type` **size** () const
- template<typename Pred > void **split** (Pred, `binomial_heap_base`< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **swap** (`left_child_next_sibling_heap`< Value\_Type, Cmp\_Fn, \_Alloc::size\_type, \_Alloc > &)
- const\_reference **top** () const

#### Protected Types

- typedef `base_type::node` **node**
- typedef `_Alloc::template rebind< left_child_next_sibling_heap_node_< Value_Type, _Alloc::size_type, _Alloc >::other` **node\_allocator**
- typedef `_Alloc::size_type` **node\_metadata**
- typedef `std::pair< node_pointer, node_pointer >` **node\_pointer\_pair**

### Protected Member Functions

- void **actual\_erase\_node** (node\_pointer)
- void **bubble\_to\_top** (node\_pointer)
- void **clear\_imp** (node\_pointer)
- template<typename It > void **copy\_from\_range** (It, It)
- void **find\_max** ()
- node\_pointer **get\_new\_node\_for\_insert** (const\_reference)
- node\_pointer **prune** (Pred)
- void **swap** (binomial\_heap\_base< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **swap\_with\_parent** (node\_pointer, node\_pointer)
- void **to\_linked\_list** ()
- void **value\_swap** (left\_child\_next\_sibling\_heap &)

### Static Protected Member Functions

- static void **make\_child\_of** (node\_pointer, node\_pointer)
- static node\_pointer **parent** (node\_pointer)

### Protected Attributes

- node\_pointer **m\_p\_max**
- node\_pointer **m\_p\_root**
- size\_type **m\_size**

#### 4.200.1 Detailed Description

template<typename Value\_Type, typename Cmp\_Fn, typename \_Alloc>class \_\_gnu\_pbds::detail::binomial\_heap< Value\_Type, Cmp\_Fn, \_Alloc >

Binomial heap.

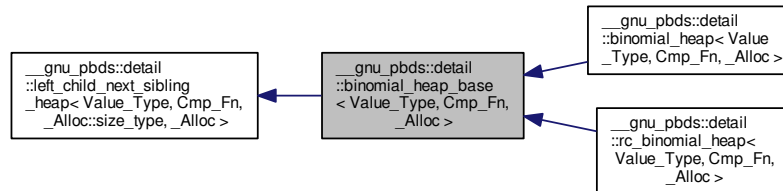
Definition at line 68 of file binomial\_heap\_.hpp.

The documentation for this class was generated from the following file:

- [binomial\\_heap\\_.hpp](#)

4.201 `__gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >`:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `__rebind_v::const_pointer` **const\_pointer**
- typedef `__rebind_v::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::point_const_iterator` **point\_const\_iterator**
- typedef `base_type::point_iterator` **point\_iterator**
- typedef `__rebind_v::pointer` **pointer**
- typedef `__rebind_v::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `Value_Type` **value\_type**

## Public Member Functions

- **iterator** **begin** ()
- **const\_iterator** **begin** () const
- void **clear** ()
- bool **empty** () const
- **iterator** **end** ()
- **const\_iterator** **end** () const
- void **erase** (**point\_iterator**)
- template<typename Pred > **size\_type** **erase\_if** (Pred)
- `Cmp_Fn` & **get\_cmp\_fn** ()
- const `Cmp_Fn` & **get\_cmp\_fn** () const
- void **join** (**binomial\_heap\_base**< `Value_Type`, `Cmp_Fn`, `_Alloc` > &)
- **size\_type** **max\_size** () const
- void **modify** (**point\_iterator**, const\_reference)
- void **pop** ()
- **point\_iterator** **push** (const\_reference)
- **size\_type** **size** () const
- template<typename Pred > void **split** (Pred, **binomial\_heap\_base**< `Value_Type`, `Cmp_Fn`, `_Alloc` > &)
- void **swap** (**left\_child\_next\_sibling\_heap**< `Value_Type`, `Cmp_Fn`, `_Alloc::size_type`, `_Alloc` > &)
- const\_reference **top** () const

### Protected Types

- typedef base\_type::node **node**
- typedef \_Alloc::template rebind< [left\\_child\\_next\\_sibling\\_heap\\_node](#)< Value\_Type, \_Alloc::size\_type, \_Alloc > >::other **node\_allocator**
- typedef base\_type::node\_const\_pointer **node\_const\_pointer**
- typedef \_Alloc::size\_type **node\_metadata**
- typedef base\_type::node\_pointer **node\_pointer**
- typedef [std::pair](#)< node\_pointer, node\_pointer > **node\_pointer\_pair**

### Protected Member Functions

- **binomial\_heap\_base** (const Cmp\_Fn &)
- **binomial\_heap\_base** (const [binomial\\_heap\\_base](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **actual\_erase\_node** (node\_pointer)
- void **bubble\_to\_top** (node\_pointer)
- void **clear\_imp** (node\_pointer)
- template<typename It > void **copy\_from\_range** (It, It)
- void **find\_max** ()
- node\_pointer **get\_new\_node\_for\_insert** (const\_reference)
- node\_pointer **prune** (Pred)
- void **swap** ([binomial\\_heap\\_base](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **swap\_with\_parent** (node\_pointer, node\_pointer)
- void **to\_linked\_list** ()
- void **value\_swap** ([left\\_child\\_next\\_sibling\\_heap](#) &)

### Static Protected Member Functions

- static void **make\_child\_of** (node\_pointer, node\_pointer)
- static node\_pointer **parent** (node\_pointer)

### Protected Attributes

- node\_pointer **m\_p\_max**
- node\_pointer **m\_p\_root**
- size\_type **m\_size**

#### 4.201.1 Detailed Description

template<typename Value\_Type, typename Cmp\_Fn, typename \_Alloc>class [\\_\\_gnu\\_pbds::detail::binomial\\_heap\\_base](#)< Value\_Type, Cmp\_Fn, \_Alloc >

Base class for binomial heap.

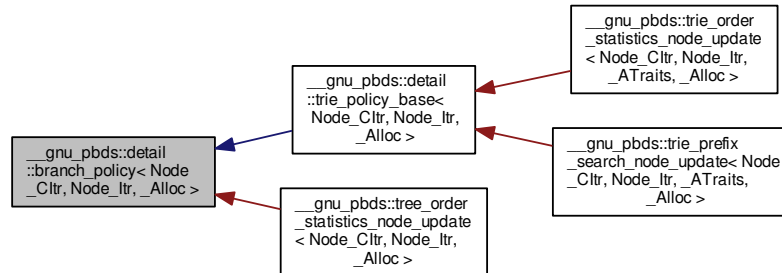
Definition at line 77 of file [binomial\\_heap\\_base\\_.hpp](#).

The documentation for this class was generated from the following file:

- [binomial\\_heap\\_base\\_.hpp](#)

## 4.202 \_\_gnu\_pbds::detail::branch\_policy&lt; Node\_Cltr, Node\_Itr, \_Alloc &gt; Struct Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::branch\_policy< Node\_Cltr, Node\_Itr, \_Alloc >:



## Protected Types

- typedef `rebind_v::const_pointer` **const\_pointer**
- typedef `rebind_v::const_reference` **const\_reference**
- typedef `Node_Itr::value_type` **it\_type**
- typedef `rebind_k::const_reference` **key\_const\_reference**
- typedef `value_type::first_type` **key\_type**
- typedef `remove_const< key_type >::type` **rckey\_type**
- typedef `remove_const< value_type >::type` **rcvalue\_type**
- typedef `_Alloc::template rebind< rkey_type >::other` **rebind\_k**
- typedef `_Alloc::template rebind< rcvalue_type >::other` **rebind\_v**
- typedef `rebind_v::reference` **reference**
- typedef `std::iterator_traits< it_type >::value_type` **value\_type**

## Protected Member Functions

- virtual `it_type` **end** ()=0
- `it_type` **end\_iterator** () const

## Static Protected Member Functions

- static `key_const_reference` **extract\_key** (const\_reference r\_val)

## 4.202.1 Detailed Description

template<typename Node\_Cltr, typename Node\_Itr, typename \_Alloc> struct \_\_gnu\_pbds::detail::branch\_policy< Node\_Cltr, Node\_Itr, \_Alloc >

Primary template, base class for branch structure policies.

Definition at line 52 of file `branch_policy.hpp`.

The documentation for this struct was generated from the following file:

- [branch\\_policy.hpp](#)

#### 4.203 `__gnu_pbds::detail::branch_policy< Node_Cltr, Node_Cltr, _Alloc >` Struct Template Reference

##### Protected Types

- `typedef rebind_v::const_pointer` **const\_pointer**
- `typedef rebind_v::const_reference` **const\_reference**
- `typedef Node_Cltr::value_type` **it\_type**
- `typedef rebind_v::const_reference` **key\_const\_reference**
- `typedef value_type` **key\_type**
- `typedef remove_const< value_type >::type` **rcvalue\_type**
- `typedef _Alloc::template rebind< rcvalue_type >::other` **rebind\_v**
- `typedef rebind_v::reference` **reference**
- `typedef std::iterator_traits< it_type >::value_type` **value\_type**

##### Protected Member Functions

- virtual `it_type` **end** () const =0
- `it_type` **end\_iterator** () const

##### Static Protected Member Functions

- static `key_const_reference` **extract\_key** (const\_reference r\_val)

##### 4.203.1 Detailed Description

`template<typename Node_Cltr, typename _Alloc>struct __gnu_pbds::detail::branch_policy< Node_Cltr, Node_Cltr, _Alloc >`

Specialization for const iterators.

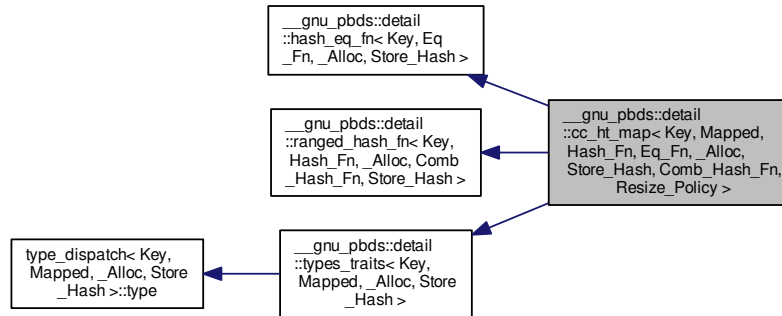
Definition at line 88 of file `branch_policy.hpp`.

The documentation for this struct was generated from the following file:

- [branch\\_policy.hpp](#)

## 4.204 `__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >`:



### Public Types

- enum { **store\_hash** }
- typedef `_Alloc` **allocator\_type**
- typedef `Comb_Hash_Fn` **comb\_hash\_fn**
- typedef `const_iterator` **const\_iterator**
- typedef `traits_base::const_pointer` **const\_pointer**
- typedef `traits_base::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `Eq_Fn` **eq\_fn**
- typedef `Hash_Fn` **hash\_fn**
- typedef `iterator` **iterator**
- typedef `traits_base::key_const_pointer` **key\_const\_pointer**
- typedef `traits_base::key_const_reference` **key\_const\_reference**
- typedef `traits_base::key_pointer` **key\_pointer**
- typedef `traits_base::key_reference` **key\_reference**
- typedef `traits_base::key_type` **key\_type**
- typedef `traits_base::mapped_const_pointer` **mapped\_const\_pointer**
- typedef `traits_base::mapped_const_reference` **mapped\_const\_reference**
- typedef `traits_base::mapped_pointer` **mapped\_pointer**
- typedef `traits_base::mapped_reference` **mapped\_reference**
- typedef `traits_base::mapped_type` **mapped\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**
- typedef `point_const_iterator` **point\_const\_iterator**
- typedef `point_iterator` **point\_iterator**
- typedef `traits_base::pointer` **pointer**
- typedef `traits_base::reference` **reference**
- typedef `Resize_Policy` **resize\_policy**
- typedef `_Alloc::size_type` **size\_type**
- typedef `integral_constant< int, Store_Hash >` **store\_extra**
- typedef `traits_base::value_type` **value\_type**



## Public Member Functions

- **cc\_ht\_map** (const Hash\_Fn &)
- **cc\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &)
- **cc\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &, const Comb\_Hash\_Fn &)
- **cc\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &, const Comb\_Hash\_Fn &, const Resize\_Policy &)
- **cc\_ht\_map** (const [cc\\_ht\\_map](#)< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Hash\_Fn, Resize\_Policy > &)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- template<typename It > void **copy\_from\_range** (It, It)
- bool [empty](#) () const
- iterator **end** ()
- const\_iterator **end** () const
- bool **erase** (key\_const\_reference)
- template<typename Pred > size\_type **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference)
- point\_const\_iterator **find** (key\_const\_reference) const
- point\_iterator **find\_end** ()
- point\_const\_iterator **find\_end** () const
- Comb\_Hash\_Fn & [get\\_comb\\_hash\\_fn](#) ()
- const Comb\_Hash\_Fn & [get\\_comb\\_hash\\_fn](#) () const
- Eq\_Fn & [get\\_eq\\_fn](#) ()
- const Eq\_Fn & [get\\_eq\\_fn](#) () const
- Hash\_Fn & [get\\_hash\\_fn](#) ()
- const Hash\_Fn & [get\\_hash\\_fn](#) () const
- Resize\_Policy & [get\\_resize\\_policy](#) ()
- const Resize\_Policy & [get\\_resize\\_policy](#) () const
- void **initialize** ()
- [std::pair](#)< point\_iterator, bool > **insert** (const\_reference r\_val)
- size\_type **max\_size** () const
- mapped\_reference **operator[]** (key\_const\_reference r\_key)
- size\_type **size** () const
- void **swap** ([cc\\_ht\\_map](#)< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Hash\_Fn, Resize\_Policy > &)

## Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

## Friends

- class **const\_iterator\_**
- class **iterator\_**

#### 4.204.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>class __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_↵Hash, Comb_Hash_Fn, Resize_Policy >
```

A collision-chaining hash-based container.

## Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Hash_Fn</i>	Hashing functor. Default is <code>__gnu_cxx::hash</code> .
<i>Eq_Fn</i>	Equal functor. Default <code>std::equal_to&lt;Key&gt;</code>
<i>_Alloc</i>	Allocator type.
<i>Store_Hash</i>	If key type stores extra metadata. Defaults to false.
<i>Comb_Hash_Fn</i>	Combining hash functor. If <i>Hash_Fn</i> is not null_type, then this is the ranged-hash functor; otherwise, this is the range-hashing functor. XXX(See Design::Hash-Based Containers::Hash Policies.) Default <code>direct_mask_range_hashing</code> .
<i>Resize_Policy</i>	Resizes hash. Defaults to <code>hash_standard_resize_policy</code> , using <code>hash_exponential_size_policy</code> and <code>hash_load_check_resize_trigger</code> .

Bases are: `detail::hash_eq_fn`, `Resize_Policy`, `detail::ranged_hash_fn`, `detail::types_traits`. (Optional: `detail::debug_map_base`.)

Definition at line 139 of file `cc_ht_map.hpp`.

## 4.204.2 Member Function Documentation

4.204.2.1 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > bool __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::empty ( ) const [inline]`

True if `size() == 0`.

Definition at line 52 of file `cc_ht_map.hpp`.

4.204.2.2 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > Comb_Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_comb_hash_fn ( )`

Return current `comb_hash_fn`.

Definition at line 70 of file `cc_ht_map.hpp`.

4.204.2.3 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > const Comb_Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_comb_hash_fn ( ) const`

Return current `const comb_hash_fn`.

Definition at line 76 of file `cc_ht_map.hpp`.

4.204.2.4 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > Eq_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_eq_fn ( )`

Return current `eq_fn`.

Definition at line 58 of file `cc_ht_map.hpp`.

4.204.2.5 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > const Eq_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_eq_fn ( ) const`

Return current `const eq_fn`.

Definition at line 64 of file `cc_ht_map.hpp`.

4.204.2.6 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_hash_fn ( )`

Return current `hash_fn`.

Definition at line 46 of file `cc_ht_map.hpp`.

4.204.2.7 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > const Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_hash_fn ( ) const`

Return current const `hash_fn`.

Definition at line 52 of file `cc_ht_map.hpp`.

4.204.2.8 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > Resize_Policy & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_resize_policy ( )`

Return current `resize_policy`.

Definition at line 82 of file `cc_ht_map.hpp`.

4.204.2.9 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > const Resize_Policy & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_resize_policy ( ) const`

Return current const `resize_policy`.

Definition at line 88 of file `cc_ht_map.hpp`.

The documentation for this class was generated from the following file:

- [cc\\_ht\\_map.hpp](#)

## 4.205 `__gnu_pbds::detail::cond_dealtor< Entry, _Alloc >` Class Template Reference

### Public Types

- typedef `HT_Map::entry` **entry**
- typedef `HT_Map::entry_allocator` **entry\_allocator**
- typedef `__rebind_e::other` **entry\_allocator**
- typedef `entry_allocator::pointer` **entry\_pointer**
- typedef `HT_Map::key_type` **key\_type**

### Public Member Functions

- **cond\_dealtor** (`entry_allocator *p_a, entry *p_e`)
- **cond\_dealtor** (`entry_pointer p_e`)
- void **set\_key\_destruct** ()
- void **set\_no\_action** ()
- void **set\_no\_action\_destructor** ()

## Protected Attributes

- bool **m\_key\_destruct**
- entry\_allocator \*const **m\_p\_a**
- entry \*const **m\_p\_e**

## 4.205.1 Detailed Description

```
template<typename Entry, typename _Alloc>class __gnu_pbds::detail::cond_dealtor< Entry, _Alloc >
```

Conditional deallocate constructor argument.

Conditional dey destructor, cc\_hash.

Definition at line 50 of file cond\_dealtor.hpp.

The documentation for this class was generated from the following files:

- [cond\\_dealtor.hpp](#)
- [cond\\_key\\_dtor\\_entry\\_dealtor.hpp](#)

## 4.206 \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, Tag, Policy\_TI > Struct Template Reference

## 4.206.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Tag, typename Policy_TI = null_type>struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, Tag, Policy_TI >
```

Dispatch mechanism, primary template for associative types.

Definition at line 449 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.207 \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp, Cmp\_Fn, \_Alloc, binary\_heap\_tag, null\_type > Struct Template Reference

## Public Types

- typedef [binary\\_heap](#)< \_VTp, Cmp\_Fn, \_Alloc > [type](#)

## 4.207.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type >
```

Specialization for binary\_heap.

Definition at line 95 of file priority\_queue\_base\_dispatch.hpp.

#### 4.207.2 Member Typedef Documentation

4.207.2.1 `template<typename _VTp, typename Cmp_Fn, typename _Alloc> typedef binary_heap<_VTp, Cmp_Fn, _Alloc>  
__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type  
>::type`

Dispatched type.

Definition at line 99 of file `priority_queue_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [priority\\_queue\\_base\\_dispatch.hpp](#)

#### 4.208 `__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type>` Struct Template Reference

##### Public Types

- typedef [binomial\\_heap](#)<\_VTp, Cmp\_Fn, \_Alloc> [type](#)

#### 4.208.1 Detailed Description

`template<typename _VTp, typename Cmp_Fn, typename _Alloc> struct __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type>`

Specialization for `binomial_heap`.

Definition at line 77 of file `priority_queue_base_dispatch.hpp`.

#### 4.208.2 Member Typedef Documentation

4.208.2.1 `template<typename _VTp, typename Cmp_Fn, typename _Alloc> typedef binomial_heap<_VTp, Cmp_Fn, _Alloc>  
__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type  
>::type`

Dispatched type.

Definition at line 81 of file `priority_queue_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [priority\\_queue\\_base\\_dispatch.hpp](#)

#### 4.209 `__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type>` Struct Template Reference

##### Public Types

- typedef [pairing\\_heap](#)<\_VTp, Cmp\_Fn, \_Alloc> [type](#)

#### 4.209.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc> struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type >
```

Specialization for pairing\_heap.

Definition at line 68 of file priority\_queue\_base\_dispatch.hpp.

#### 4.209.2 Member Typedef Documentation

4.209.2.1 `template<typename _VTp, typename Cmp_Fn, typename _Alloc > typedef pairing_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type >::type`

Dispatched type.

Definition at line 72 of file priority\_queue\_base\_dispatch.hpp.

The documentation for this struct was generated from the following file:

- [priority\\_queue\\_base\\_dispatch.hpp](#)

### 4.210 `__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >` Struct Template Reference

#### Public Types

- typedef `rc_binomial_heap< _VTp, Cmp_Fn, _Alloc > type`

#### 4.210.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc> struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >
```

Specialization for rc\_binary\_heap.

Definition at line 86 of file priority\_queue\_base\_dispatch.hpp.

#### 4.210.2 Member Typedef Documentation

4.210.2.1 `template<typename _VTp, typename Cmp_Fn, typename _Alloc > typedef rc_binomial_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >::type`

Dispatched type.

Definition at line 90 of file priority\_queue\_base\_dispatch.hpp.

The documentation for this struct was generated from the following file:

- [priority\\_queue\\_base\\_dispatch.hpp](#)

#### 4.211 `__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type >` Struct Template Reference

##### Public Types

- typedef `thin_heap< _VTp, Cmp_Fn, _Alloc >` [type](#)

##### 4.211.1 Detailed Description

`template<typename _VTp, typename Cmp_Fn, typename _Alloc>struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type >`

Specialization for `thin_heap`.

Definition at line 104 of file `priority_queue_base_dispatch.hpp`.

##### 4.211.2 Member Typedef Documentation

4.211.2.1 `template<typename _VTp, typename Cmp_Fn, typename _Alloc > typedef thin_heap< _VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type >::type`

Dispatched type.

Definition at line 108 of file `priority_queue_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [priority\\_queue\\_base\\_dispatch.hpp](#)

#### 4.212 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_TI >` Struct Template Reference

##### Public Types

- typedef `cc_ht_map< Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at2t >` [type](#)

##### 4.212.1 Detailed Description

`template<typename Key, typename Mapped, typename _Alloc, typename Policy_TI>struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_TI >`

Specialization collision-chaining hash map.

Definition at line 258 of file `container_base_dispatch.hpp`.

##### 4.212.2 Member Typedef Documentation

4.212.2.1 `template<typename Key, typename Mapped, typename _Alloc, typename Policy_TI > typedef cc_ht_map<Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at2t > __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_TI >::type`

Dispatched type.



Definition at line 275 of file container\_base\_dispatch.hpp.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

#### 4.213 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl >` Struct Template Reference

##### Public Types

- typedef [gp\\_ht\\_map](#)< Key, Mapped, at0t, at1t, \_Alloc, at3t::value, at4t, at5t, at2t > [type](#)

##### 4.213.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl >
```

Specialization general-probe hash map.

Definition at line 303 of file container\_base\_dispatch.hpp.

##### 4.213.2 Member Typedef Documentation

4.213.2.1 `template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl > typedef gp_ht_map<Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t> __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl >::type`

Dispatched type.

Definition at line 322 of file container\_base\_dispatch.hpp.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

#### 4.214 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl >` Struct Template Reference

##### Public Types

- typedef [lu\\_map](#)< Key, Mapped, at0t, \_Alloc, at1t > [type](#)

##### 4.214.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl >
```

Specialization for list-update map.

Definition at line 107 of file container\_base\_dispatch.hpp.

#### 4.214.2 Member Typedef Documentation

4.214.2.1 `template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl > typedef lu_map<Key, Mapped, at0t, _Alloc, at1t> __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl >::type`

Dispatched type.

Definition at line 118 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

### 4.215 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >` Struct Template Reference

#### Public Types

- typedef `ov_tree_map< Key, Mapped, at0t, at1t, _Alloc >` `type`

#### 4.215.1 Detailed Description

`template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >`

Specialization ordered-vector tree map.

Definition at line 227 of file `container_base_dispatch.hpp`.

#### 4.215.2 Member Typedef Documentation

4.215.2.1 `template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl > typedef ov_tree_map<Key, Mapped, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >::type`

Dispatched type.

Definition at line 237 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

### 4.216 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_Tl >` Struct Template Reference

#### Public Types

- typedef `pat_trie_map< Key, Mapped, at1t, _Alloc >` `type`

#### 4.216.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_TI>struct __gnu_pbds::detail::container_base_↵
dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_TI >
```

Specialization for PATRICIA trie map.

Definition at line 139 of file container\_base\_dispatch.hpp.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

#### 4.217 \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, rb\_tree\_tag, Policy\_TI > Struct Template Reference

##### Public Types

- typedef [rb\\_tree\\_map](#)< Key, Mapped, at0t, at1t, \_Alloc > [type](#)

#### 4.217.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_TI>struct __gnu_pbds::detail::container_base_↵
dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_TI >
```

Specialization for R-B tree map.

Definition at line 165 of file container\_base\_dispatch.hpp.

#### 4.217.2 Member Typedef Documentation

```
4.217.2.1 template<typename Key , typename Mapped , typename _Alloc , typename Policy_TI > typedef rb_tree_map<Key,
Mapped, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag,
Policy_TI >::type
```

Dispatched type.

Definition at line 175 of file container\_base\_dispatch.hpp.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

#### 4.218 \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, splay\_tree\_tag, Policy\_TI > Struct Template Reference

##### Public Types

- typedef [splay\\_tree\\_map](#)< Key, Mapped, at0t, at1t, \_Alloc > [type](#)

#### 4.218.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_TI>struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_TI >
```

Specialization splay tree map.

Definition at line 195 of file `container_base_dispatch.hpp`.

#### 4.218.2 Member Typedef Documentation

4.218.2.1 `template<typename Key , typename Mapped , typename _Alloc , typename Policy_TI > typedef splay_tree_map<Key, Mapped, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_TI >::type`

Dispatched type.

Definition at line 206 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

### 4.219 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_TI >` Struct Template Reference

#### Public Types

- `typedef cc_ht_set< Key, null\_type, at0t, at1t, _Alloc, at3t::value, at4t, at2t > type`

#### 4.219.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_TI>struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_TI >
```

Specialization collision-chaining hash set.

Definition at line 280 of file `container_base_dispatch.hpp`.

#### 4.219.2 Member Typedef Documentation

4.219.2.1 `template<typename Key , typename _Alloc , typename Policy_TI > typedef cc_ht_set<Key, null_type, at0t, at1t, _Alloc, at3t::value, at4t, at2t> __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_TI >::type`

Dispatched type.

Definition at line 298 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

## 4.220 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >` Struct Template Reference

### Public Types

- typedef `gp_ht_set< Key, null\_type, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t >` [type](#)

### 4.220.1 Detailed Description

`template<typename Key, typename _Alloc, typename Policy_Tl>struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >`

Specialization general-probe hash set.

Definition at line 327 of file `container_base_dispatch.hpp`.

### 4.220.2 Member Typedef Documentation

4.220.2.1 `template<typename Key , typename _Alloc , typename Policy_Tl > typedef gp_ht_set<Key, null_type, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t> __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >::type`

Dispatched type.

Definition at line 347 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

## 4.221 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_Tl >` Struct Template Reference

### Public Types

- typedef `lu_set< Key, null\_type, at0t, _Alloc, at1t >` [type](#)

### 4.221.1 Detailed Description

`template<typename Key, typename _Alloc, typename Policy_Tl>struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_Tl >`

Specialization for list-update set.

Definition at line 123 of file `container_base_dispatch.hpp`.

### 4.221.2 Member Typedef Documentation

4.221.2.1 `template<typename Key , typename _Alloc , typename Policy_Tl > typedef lu_set<Key, null_type, at0t, _Alloc, at1t> __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_Tl >::type`

Dispatched type.

Definition at line 134 of file container\_base\_dispatch.hpp.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

## **4.222 \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, ov\_tree\_tag, Policy\_Tl > Struct Template Reference**

### **Public Types**

- typedef ov\_tree\_set< Key, [null\\_type](#), at0t, at1t, \_Alloc > [type](#)

#### **4.222.1 Detailed Description**

template<typename Key, typename \_Alloc, typename Policy\_Tl>struct \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, ov\_tree\_tag, Policy\_Tl >

Specialization ordered-vector tree set.

Definition at line 242 of file container\_base\_dispatch.hpp.

#### **4.222.2 Member Typedef Documentation**

4.222.2.1 template<typename Key , typename \_Alloc , typename Policy\_Tl > typedef ov\_tree\_set<Key, [null\\_type](#), at0t, at1t, \_Alloc> \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, [null\\_type](#), \_Alloc, ov\_tree\_tag, Policy\_Tl >::[type](#)

Dispatched type.

Definition at line 253 of file container\_base\_dispatch.hpp.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

## **4.223 \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, pat\_trie\_tag, Policy\_Tl > Struct Template Reference**

### **Public Types**

- typedef pat\_trie\_set< Key, [null\\_type](#), at1t, \_Alloc > [type](#)

#### **4.223.1 Detailed Description**

template<typename Key, typename \_Alloc, typename Policy\_Tl>struct \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, pat\_trie\_tag, Policy\_Tl >

Specialization for PATRICIA trie set.

Definition at line 151 of file container\_base\_dispatch.hpp.

#### 4.223.2 Member Typedef Documentation

4.223.2.1 `template<typename Key , typename _Alloc , typename Policy_TI > typedef pat_trie_set<Key, null_type, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_TI >::type`

Dispatched type.

Definition at line 160 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

#### 4.224 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_TI >` Struct Template Reference

##### Public Types

- `typedef rb_tree_set< Key, null_type, at0t, at1t, _Alloc > type`

##### 4.224.1 Detailed Description

`template<typename Key, typename _Alloc, typename Policy_TI> struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_TI >`

Specialization for R-B tree set.

Definition at line 180 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

#### 4.225 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_TI >` Struct Template Reference

##### Public Types

- `typedef splay_tree_set< Key, null_type, at0t, at1t, _Alloc > type`

##### 4.225.1 Detailed Description

`template<typename Key, typename _Alloc, typename Policy_TI> struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_TI >`

Specialization splay tree set.

Definition at line 211 of file `container_base_dispatch.hpp`.

#### 4.225.2 Member Typedef Documentation

4.225.2.1 `template<typename Key , typename _Alloc , typename Policy_Tl > typedef splay_tree_set<Key, null_type, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >::type`

Dispatched type.

Definition at line 222 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

## 4.226 `__gnu_pbds::detail::default_comb_hash_fn` Struct Reference

### Public Types

- typedef [direct\\_mask\\_range\\_hashing](#) type

### 4.226.1 Detailed Description

Primary template, `default_comb_hash_fn`.

Definition at line 80 of file `standard_policies.hpp`.

### 4.226.2 Member Typedef Documentation

4.226.2.1 `typedef direct_mask_range_hashing __gnu_pbds::detail::default_comb_hash_fn::type`

Dispatched type.

Definition at line 83 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

## 4.227 `__gnu_pbds::detail::default_eq_fn< Key >` Struct Template Reference

### Public Types

- typedef [std::equal\\_to< Key >](#) type

### 4.227.1 Detailed Description

`template<typename Key>struct __gnu_pbds::detail::default_eq_fn< Key >`

Primary template, `default_eq_fn`.

Definition at line 67 of file `standard_policies.hpp`.



#### 4.227.2 Member Typedef Documentation

##### 4.227.2.1 `template<typename Key> typedef std::equal_to<Key> __gnu_pbds::detail::default_eq_fn< Key >::type`

Dispatched type.

Definition at line 70 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

#### 4.228 `__gnu_pbds::detail::default_hash_fn< Key >` Struct Template Reference

##### Public Types

- `typedef std::tr1::hash< Key > type`

##### 4.228.1 Detailed Description

```
template<typename Key>struct __gnu_pbds::detail::default_hash_fn< Key >
```

Primary template, `default_hash_fn`.

Definition at line 59 of file `standard_policies.hpp`.

#### 4.228.2 Member Typedef Documentation

##### 4.228.2.1 `template<typename Key> typedef std::tr1::hash<Key> __gnu_pbds::detail::default_hash_fn< Key >::type`

Dispatched type.

Definition at line 62 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

#### 4.229 `__gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >` Struct Template Reference

##### Public Types

- `typedef cond_type::__type type`

##### 4.229.1 Detailed Description

```
template<typename Comb_Probe_Fn>struct __gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >
```

Primary template, `default_probe_fn`.

Definition at line 117 of file `standard_policies.hpp`.

## 4.229.2 Member Typedef Documentation

4.229.2.1 `template<typename Comb_Probe_Fn > typedef cond_type::__type __gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >::type`

Dispatched type.

Definition at line 129 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

4.230 `__gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >` Struct Template Reference

## Public Types

- typedef `hash_standard_resize_policy< size_policy_type, trigger, false, size_type >` `type`

## 4.230.1 Detailed Description

`template<typename Comb_Hash_Fn> struct __gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >`

Primary template, `default_resize_policy`.

Definition at line 88 of file `standard_policies.hpp`.

## 4.230.2 Member Typedef Documentation

4.230.2.1 `template<typename Comb_Hash_Fn> typedef hash_standard_resize_policy<size_policy_type, trigger, false, size_type> __gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >::type`

Dispatched type.

Definition at line 105 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

4.231 `__gnu_pbds::detail::default_trie_access_traits< Key >` Struct Template Reference

## 4.231.1 Detailed Description

`template<typename Key> struct __gnu_pbds::detail::default_trie_access_traits< Key >`

Primary template, `default_trie_access_traits`.

Definition at line 135 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

#### 4.232 `__gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > > Struct Template Reference`

##### Public Types

- typedef [trie\\_string\\_access\\_traits< string\\_type > type](#)

##### 4.232.1 Detailed Description

template<typename Char, typename Char\_Traits>struct `__gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >`

Partial specialization, `default_trie_access_traits`.

Definition at line 142 of file `standard_policies.hpp`.

##### 4.232.2 Member Typedef Documentation

4.232.2.1 `template<typename Char , typename Char_Traits > typedef trie_string_access_traits<string_type> __gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >::type`

Dispatched type.

Definition at line 149 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

#### 4.233 `__gnu_pbds::detail::default_update_policy Struct Reference`

##### Public Types

- typedef [lu\\_move\\_to\\_front\\_policy type](#)

##### 4.233.1 Detailed Description

Default update policy.

Definition at line 109 of file `standard_policies.hpp`.

##### 4.233.2 Member Typedef Documentation

4.233.2.1 `typedef lu_move_to_front_policy __gnu_pbds::detail::default_update_policy::type`

Dispatched type.

Definition at line 112 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

4.234 `__gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc >` Struct Template Reference

## Public Types

- typedef `const_iterator` **const\_reference**
- typedef `const_reference` **reference**
- typedef `const_iterator` **value\_type**

## 4.234.1 Detailed Description

`template<typename Key, typename Data, typename _Alloc>struct __gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc >`

Constant node iterator.

Definition at line 52 of file `null_node_metadata.hpp`.

The documentation for this struct was generated from the following file:

- [null\\_node\\_metadata.hpp](#)

4.235 `__gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, No_Throw >` Struct Template Reference

## 4.235.1 Detailed Description

`template<typename _VTp, typename Cmp_Fn, typename _Alloc, bool No_Throw>struct __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, No_Throw >`

Entry compare, primary template.

Definition at line 50 of file `entry_cmp.hpp`.

The documentation for this struct was generated from the following file:

- [entry\\_cmp.hpp](#)

4.236 `__gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >` Struct Template Reference

## Classes

- struct [type](#)

## Public Types

- typedef `__rebind_v::other::const_pointer` **entry**

## 4.236.1 Detailed Description

`template<typename _VTp, typename Cmp_Fn, typename _Alloc>struct __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >`

Specialization, false.

Definition at line 62 of file entry\_cmp.hpp.

The documentation for this struct was generated from the following file:

- [entry\\_cmp.hpp](#)

#### 4.237 `__gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, false >::type` Struct Reference

Inherits `Cmp_Fn`.

##### Public Member Functions

- **type** (const `Cmp_Fn` &other)
- bool **operator()** (entry lhs, entry rhs) const

##### 4.237.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>struct __gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, false>::type
```

Compare plus entry.

Definition at line 71 of file entry\_cmp.hpp.

The documentation for this struct was generated from the following file:

- [entry\\_cmp.hpp](#)

#### 4.238 `__gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, true >` Struct Template Reference

##### Public Types

- typedef `Cmp_Fn` [type](#)

##### 4.238.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>struct __gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, true>
```

Specialization, true.

Definition at line 54 of file entry\_cmp.hpp.

##### 4.238.2 Member Typedef Documentation

4.238.2.1 `template<typename _VTp, typename Cmp_Fn, typename _Alloc> typedef Cmp_Fn __gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, true >::type`

Compare.

Definition at line 57 of file entry\_cmp.hpp.

The documentation for this struct was generated from the following file:

- [entry\\_cmp.hpp](#)

## 4.239 `__gnu_pbds::detail::entry_pred<_VTp, Pred, _Alloc, No_Throw>` Struct Template Reference

### 4.239.1 Detailed Description

`template<typename _VTp, typename Pred, typename _Alloc, bool No_Throw> struct __gnu_pbds::detail::entry_pred<_VTp, Pred, _Alloc, No_Throw>`

Entry predicate primary class template.

Definition at line 50 of file `entry_pred.hpp`.

The documentation for this struct was generated from the following file:

- [entry\\_pred.hpp](#)

## 4.240 `__gnu_pbds::detail::entry_pred<_VTp, Pred, _Alloc, false>` Struct Template Reference

### Public Types

- `typedef __rebind_v::other::const_pointer` **entry**

### 4.240.1 Detailed Description

`template<typename _VTp, typename Pred, typename _Alloc> struct __gnu_pbds::detail::entry_pred<_VTp, Pred, _Alloc, false>`

Specialization, `false`.

Definition at line 61 of file `entry_pred.hpp`.

The documentation for this struct was generated from the following file:

- [entry\\_pred.hpp](#)

## 4.241 `__gnu_pbds::detail::entry_pred<_VTp, Pred, _Alloc, true>` Struct Template Reference

### Public Types

- `typedef Pred` **type**

### 4.241.1 Detailed Description

`template<typename _VTp, typename Pred, typename _Alloc> struct __gnu_pbds::detail::entry_pred<_VTp, Pred, _Alloc, true>`

Specialization, `true`.

Definition at line 54 of file `entry_pred.hpp`.

The documentation for this struct was generated from the following file:

- [entry\\_pred.hpp](#)

#### 4.242 `__gnu_pbds::detail::eq_by_less< Key, Cmp_Fn >` Struct Template Reference

Inherits `Cmp_Fn`.

##### Public Member Functions

- `bool operator()` (`const Key &r_lhs, const Key &r_rhs`) `const`

##### 4.242.1 Detailed Description

`template<typename Key, class Cmp_Fn>struct __gnu_pbds::detail::eq_by_less< Key, Cmp_Fn >`

Equivalence function.

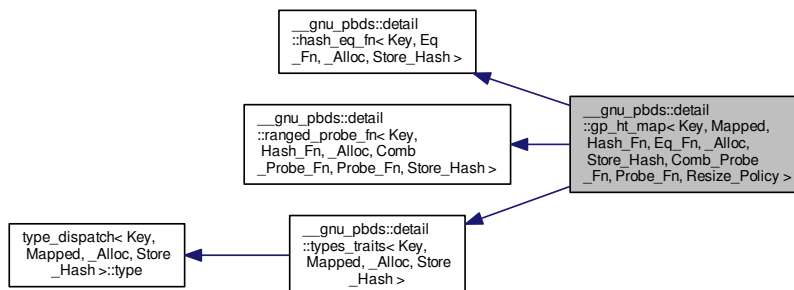
Definition at line 56 of file `eq_by_less.hpp`.

The documentation for this struct was generated from the following file:

- [eq\\_by\\_less.hpp](#)

#### 4.243 `__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >`:



##### Public Types

- `enum { store_hash }`
- `typedef _Alloc allocator_type`
- `typedef Comb_Probe_Fn comb_probe_fn`
- `typedef const\_iterator const_iterator`
- `typedef traits_base::const_pointer const_pointer`
- `typedef traits_base::const_reference const_reference`
- `typedef _Alloc::difference_type difference_type`
- `typedef Eq_Fn eq_fn`

- typedef Hash\_Fn **hash\_fn**
- typedef [iterator](#) **iterator**
- typedef traits\_base::key\_const\_pointer **key\_const\_pointer**
- typedef traits\_base::key\_const\_reference **key\_const\_reference**
- typedef traits\_base::key\_pointer **key\_pointer**
- typedef traits\_base::key\_reference **key\_reference**
- typedef traits\_base::key\_type **key\_type**
- typedef traits\_base::mapped\_const\_pointer **mapped\_const\_pointer**
- typedef traits\_base::mapped\_const\_reference **mapped\_const\_reference**
- typedef traits\_base::mapped\_pointer **mapped\_pointer**
- typedef traits\_base::mapped\_reference **mapped\_reference**
- typedef traits\_base::mapped\_type **mapped\_type**
- typedef \_\_nothrowcopy::indicator **no\_throw\_indicator**
- typedef [point\\_const\\_iterator](#) **point\_const\_iterator**
- typedef [point\\_iterator](#) **point\_iterator**
- typedef traits\_base::pointer **pointer**
- typedef Probe\_Fn **probe\_fn**
- typedef traits\_base::reference **reference**
- typedef Resize\_Policy **resize\_policy**
- typedef \_Alloc::size\_type **size\_type**
- typedef integral\_constant< int, Store\_Hash > **store\_extra**
- typedef traits\_base::value\_type **value\_type**

#### Public Member Functions

- **gp\_ht\_map** (const [gp\\_ht\\_map](#)< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy > &)
- **gp\_ht\_map** (const Hash\_Fn &)
- **gp\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &)
- **gp\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &, const Comb\_Probe\_Fn &)
- **gp\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &, const Comb\_Probe\_Fn &, const Probe\_Fn &)
- **gp\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &, const Comb\_Probe\_Fn &, const Probe\_Fn &, const Resize\_Policy &)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- template<typename It > void **copy\_from\_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- bool **erase** (key\_const\_reference)
- template<typename Pred > size\_type **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference)
- point\_const\_iterator **find** (key\_const\_reference) const
- point\_iterator **find\_end** ()
- point\_const\_iterator **find\_end** () const
- Comb\_Probe\_Fn & **get\_comb\_probe\_fn** ()
- const Comb\_Probe\_Fn & **get\_comb\_probe\_fn** () const
- Eq\_Fn & **get\_eq\_fn** ()
- const Eq\_Fn & **get\_eq\_fn** () const



- Hash\_Fn & [get\\_hash\\_fn](#) ()
- const Hash\_Fn & [get\\_hash\\_fn](#) () const
- Probe\_Fn & [get\\_probe\\_fn](#) ()
- const Probe\_Fn & [get\\_probe\\_fn](#) () const
- Resize\_Policy & [get\\_resize\\_policy](#) ()
- const Resize\_Policy & [get\\_resize\\_policy](#) () const
- [std::pair](#)< point\_iterator, bool > [insert](#) (const\_reference r\_val)
- size\_type [max\\_size](#) () const
- mapped\_reference [operator\[\]](#) (key\_const\_reference r\_key)
- size\_type [size](#) () const
- void [swap](#) ([gp\\_ht\\_map](#)< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy > &)

#### Public Attributes

- no\_throw\_indicator [m\\_no\\_throw\\_copies\\_indicator](#)
- store\_extra [m\\_store\\_extra\\_indicator](#)

#### Friends

- class [const\\_iterator\\_](#)
- class [iterator\\_](#)

#### 4.243.1 Detailed Description

`template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>class __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >`

A general-probing hash-based container.

#### Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Hash_Fn</i>	Hashing functor. Default is <code>__gnu_cxx::hash</code> .
<i>Eq_Fn</i>	Equal functor. Default <code>std::equal_to&lt;Key&gt;</code>
<i>_Alloc</i>	Allocator type.
<i>Store_Hash</i>	If key type stores extra metadata. Defaults to false.
<i>Comb_Probe_Fn</i>	Combining probe functor. If <i>Hash_Fn</i> is not null_type, then this is the ranged-probe functor; otherwise, this is the range-hashing functor. XXX See Design::Hash-Based Containers::Hash Policies. Default <code>direct_mask_range_hashing</code> .
<i>Probe_Fn</i>	Probe functor. Defaults to <code>linear_probe_fn</code> , also <code>quadratic_probe_fn</code> .
<i>Resize_Policy</i>	Resizes hash. Defaults to <code>hash_standard_resize_policy</code> , using <code>hash_exponential_size_policy</code> and <code>hash_load_check_resize_trigger</code> .

Bases are: `detail::hash_eq_fn`, `Resize_Policy`, `detail::ranged_probe_fn`, `detail::types_traits`. (Optional: `detail::debug_map_base`.)

Definition at line 142 of file `gp_ht_map.hpp`.

#### 4.243.2 Member Function Documentation

4.243.2.1 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > bool __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::empty ( ) const` `[inline]`

True if size() == 0.

Definition at line 58 of file `gp_ht_map.hpp`.

4.243.2.2 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > Comb_Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_comb_probe_fn ( )`

Return current `comb_probe_fn`.

Definition at line 82 of file `gp_ht_map.hpp`.

4.243.2.3 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > const Comb_Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_comb_probe_fn ( ) const`

Return current `const comb_probe_fn`.

Definition at line 88 of file `gp_ht_map.hpp`.

4.243.2.4 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > Eq_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_eq_fn ( )`

Return current `eq_fn`.

Definition at line 58 of file `gp_ht_map.hpp`.

4.243.2.5 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > const Eq_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_eq_fn ( ) const`

Return current `const eq_fn`.

Definition at line 64 of file `gp_ht_map.hpp`.

4.243.2.6 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > Hash_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_hash_fn ( )`

Return current `hash_fn`.

Definition at line 46 of file `gp_ht_map.hpp`.

4.243.2.7 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > const Hash_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_hash_fn ( ) const`

Return current const hash\_fn.

Definition at line 52 of file gp\_ht\_map.hpp.

4.243.2.8 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_probe_fn ( )`

Return current probe\_fn.

Definition at line 70 of file gp\_ht\_map.hpp.

4.243.2.9 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > const Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_probe_fn ( ) const`

Return current const probe\_fn.

Definition at line 76 of file gp\_ht\_map.hpp.

4.243.2.10 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > Resize_Policy & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_resize_policy ( )`

Return current resize\_policy.

Definition at line 94 of file gp\_ht\_map.hpp.

4.243.2.11 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > const Resize_Policy & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_resize_policy ( ) const`

Return current const resize\_policy.

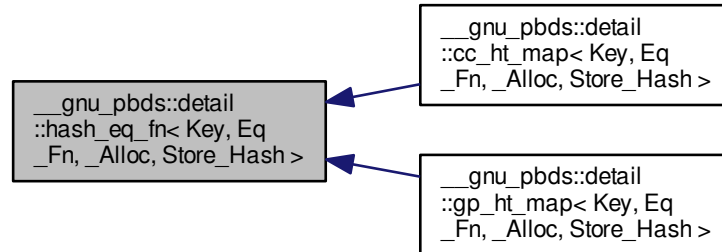
Definition at line 100 of file gp\_ht\_map.hpp.

The documentation for this class was generated from the following file:

- [gp\\_ht\\_map.hpp](#)

4.244 `__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash >`:



## 4.244.1 Detailed Description

`template<typename Key, typename Eq_Fn, typename _Alloc, bool Store_Hash>struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash >`

Primary template.

Definition at line 54 of file `hash_eq_fn.hpp`.

The documentation for this struct was generated from the following file:

- [hash\\_eq\\_fn.hpp](#)

4.245 `__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false >` Struct Template Reference

Inherits `Eq_Fn`.

## Public Types

- typedef `Eq_Fn` **eq\_fn\_base**
- typedef `_Alloc::template rebind< Key >::other` **key\_allocator**
- typedef `key_allocator::const_reference` **key\_const\_reference**

## Public Member Functions

- **hash\_eq\_fn** (`const Eq_Fn &r_eq_fn`)
- bool **operator()** (`key_const_reference r_lhs_key, key_const_reference r_rhs_key`) const
- void **swap** (`const hash\_eq\_fn &other`)

#### 4.245.1 Detailed Description

```
template<typename Key, typename Eq_Fn, typename _Alloc>struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false >
```

Specialization 1 - The client requests that hash values not be stored.

Definition at line 58 of file hash\_eq\_fn.hpp.

The documentation for this struct was generated from the following file:

- [hash\\_eq\\_fn.hpp](#)

#### 4.246 \_\_gnu\_pbds::detail::hash\_eq\_fn< Key, Eq\_Fn, \_Alloc, true > Struct Template Reference

Inherits Eq\_Fn.

##### Public Types

- typedef Eq\_Fn **eq\_fn\_base**
- typedef \_Alloc::template rebind< Key >::other **key\_allocator**
- typedef key\_allocator::const\_reference **key\_const\_reference**
- typedef \_Alloc::size\_type **size\_type**

##### Public Member Functions

- **hash\_eq\_fn** (const Eq\_Fn &r\_eq\_fn)
- bool **operator()** (key\_const\_reference r\_lhs\_key, size\_type lhs\_hash, key\_const\_reference r\_rhs\_key, size\_type rhs\_hash) const
- void **swap** (const [hash\\_eq\\_fn](#) &other)

#### 4.246.1 Detailed Description

```
template<typename Key, class Eq_Fn, class _Alloc>struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true >
```

Specialization 2 - The client requests that hash values be stored.

Definition at line 81 of file hash\_eq\_fn.hpp.

The documentation for this struct was generated from the following file:

- [hash\\_eq\\_fn.hpp](#)

#### 4.247 \_\_gnu\_pbds::detail::hash\_load\_check\_resize\_trigger\_size\_base< Size\_Type, Hold\_Size > Class Template Reference

##### 4.247.1 Detailed Description

```
template<typename Size_Type, bool Hold_Size>class __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size >
```

Primary template.

Definition at line 50 of file `hash_load_check_resize_trigger_size_base.hpp`.

The documentation for this class was generated from the following file:

- [hash\\_load\\_check\\_resize\\_trigger\\_size\\_base.hpp](#)

## 4.248 `__gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true >` Class Template Reference

### Protected Types

- typedef `Size_Type` **size\_type**

### Protected Member Functions

- `size_type` **get\_size** () const
- void **set\_size** (`size_type` size)
- void **swap** ([hash\\_load\\_check\\_resize\\_trigger\\_size\\_base](#) &other)

### 4.248.1 Detailed Description

`template<typename Size_Type>class __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true >`

Specializations.

Definition at line 54 of file `hash_load_check_resize_trigger_size_base.hpp`.

The documentation for this class was generated from the following file:

- [hash\\_load\\_check\\_resize\\_trigger\\_size\\_base.hpp](#)

## 4.249 `__gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >` Class Template Reference

Inherits `Cmp_Fn`.

### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef [left\\_child\\_next\\_sibling\\_heap\\_const\\_iterator](#)< `node`, `_Alloc` > **const\_iterator**
- typedef `__rebind_v::other::const_pointer` **const\_pointer**
- typedef `__rebind_v::other::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef [const\\_iterator](#) **iterator**
- typedef [left\\_child\\_next\\_sibling\\_heap\\_node\\_point\\_const\\_iterator](#)< `node`, `_Alloc` > **point\_const\_iterator**
- typedef [point\\_const\\_iterator](#) **point\_iterator**
- typedef `__rebind_v::other::pointer` **pointer**
- typedef `__rebind_v::other::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `Value_Type` **value\_type**

### Public Member Functions

- **left\_child\_next\_sibling\_heap** (const Cmp\_Fn &)
- **left\_child\_next\_sibling\_heap** (const [left\\_child\\_next\\_sibling\\_heap](#) &)
- [iterator](#) **begin** ()
- [const\\_iterator](#) **begin** () const
- void **clear** ()
- bool **empty** () const
- [iterator](#) **end** ()
- [const\\_iterator](#) **end** () const
- Cmp\_Fn & **get\_cmp\_fn** ()
- const Cmp\_Fn & **get\_cmp\_fn** () const
- size\_type **max\_size** () const
- size\_type **size** () const
- void **swap** ([left\\_child\\_next\\_sibling\\_heap](#)< Value\_Type, Cmp\_Fn, Node\_Metadata, \_Alloc > &)

### Protected Types

- typedef node\_allocator::value\_type **node**
- typedef \_Alloc::template rebind< [left\\_child\\_next\\_sibling\\_heap\\_node](#) < Value\_Type, Node\_Metadata, \_Alloc > >::other **node\_allocator**
- typedef node\_allocator::const\_pointer **node\_const\_pointer**
- typedef Node\_Metadata **node\_metadata**
- typedef node\_allocator::pointer **node\_pointer**
- typedef [std::pair](#)< node\_pointer, node\_pointer > **node\_pointer\_pair**

### Protected Member Functions

- void **actual\_erase\_node** (node\_pointer)
- void **bubble\_to\_top** (node\_pointer)
- void **clear\_imp** (node\_pointer)
- node\_pointer **get\_new\_node\_for\_insert** (const\_reference)
- template<typename Pred > node\_pointer **prune** (Pred)
- void **swap\_with\_parent** (node\_pointer, node\_pointer)
- void **to\_linked\_list** ()
- void **value\_swap** ([left\\_child\\_next\\_sibling\\_heap](#) &)

### Static Protected Member Functions

- static void **make\_child\_of** (node\_pointer, node\_pointer)
- static node\_pointer **parent** (node\_pointer)

### Protected Attributes

- node\_pointer **m\_p\_root**
- size\_type **m\_size**

## 4.249.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename Node_Metadata, typename _Alloc>class __gnu_pbds::detail::left_↵
child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >
```

Base class for a basic heap.

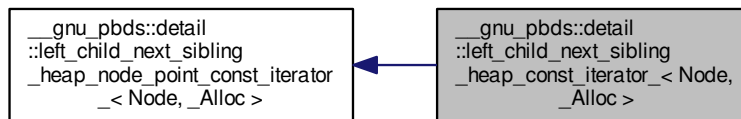
Definition at line 90 of file `left_child_next_sibling_heap_.hpp`.

The documentation for this class was generated from the following file:

- [left\\_child\\_next\\_sibling\\_heap\\_.hpp](#)

4.250 `__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >`:



## Public Types

- typedef `base_type::const_pointer` `const_pointer`
- typedef `base_type::const_reference` `const_reference`
- typedef `_Alloc::difference_type` `difference_type`
- typedef `std::forward_iterator_tag` `iterator_category`
- typedef `base_type::pointer` `pointer`
- typedef `base_type::reference` `reference`
- typedef `base_type::value_type` `value_type`

## Public Member Functions

- `left_child_next_sibling_heap_const_iterator_` (`node_pointer p_nd`)
- `left_child_next_sibling_heap_const_iterator_` ()
- `left_child_next_sibling_heap_const_iterator_` (`const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > &other`)
- `bool operator!=` (`const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > &other`) `const`
- `bool operator!=` (`const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other`) `const`
- `const_reference operator*` () `const`
- `left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & operator++` ()
- `left_child_next_sibling_heap_const_iterator_< Node, _Alloc > operator++` (`int`)
- `const_pointer operator->` () `const`
- `bool operator==` (`const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > &other`) `const`
- `bool operator==` (`const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other`) `const`



## Public Attributes

- node\_pointer **m\_p\_nd**

## 4.250.1 Detailed Description

```
template<typename Node, typename _Alloc>class __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc
>
```

Const point-type iterator.

Definition at line 60 of file left\_child\_next\_sibling\_heap\_/const\_iterator.hpp.

## 4.250.2 Member Typedef Documentation

```
4.250.2.1 template<typename Node , typename _Alloc > typedef base_type::const_pointer
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::const_pointer
```

Iterator's const pointer type.

Definition at line 81 of file left\_child\_next\_sibling\_heap\_/const\_iterator.hpp.

```
4.250.2.2 template<typename Node , typename _Alloc > typedef base_type::const_reference
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::const_reference
```

Iterator's const reference type.

Definition at line 87 of file left\_child\_next\_sibling\_heap\_/const\_iterator.hpp.

```
4.250.2.3 template<typename Node , typename _Alloc > typedef _Alloc::difference_type __gnu_pbds::detail::left_child_↵
next_sibling_heap_const_iterator_< Node, _Alloc >::difference_type
```

Difference type.

Definition at line 72 of file left\_child\_next\_sibling\_heap\_/const\_iterator.hpp.

```
4.250.2.4 template<typename Node , typename _Alloc > typedef std::forward_iterator_tag
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::iterator_category
```

Category.

Definition at line 69 of file left\_child\_next\_sibling\_heap\_/const\_iterator.hpp.

```
4.250.2.5 template<typename Node , typename _Alloc > typedef base_type::pointer __gnu_pbds::detail::left_child_↵
next_sibling_heap_const_iterator_< Node, _Alloc >::pointer
```

Iterator's pointer type.

Definition at line 78 of file left\_child\_next\_sibling\_heap\_/const\_iterator.hpp.

```
4.250.2.6 template<typename Node , typename _Alloc > typedef base_type::reference __gnu_pbds::detail::left_child_↵
next_sibling_heap_const_iterator_< Node, _Alloc >::reference
```

Iterator's reference type.

Definition at line 84 of file left\_child\_next\_sibling\_heap\_/const\_iterator.hpp.

```
4.250.2.7 template<typename Node , typename _Alloc > typedef base_type::value_type
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::value_type
```

Iterator's value type.

Definition at line 75 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

#### 4.250.3 Constructor & Destructor Documentation

```
4.250.3.1 template<typename Node , typename _Alloc > __gnu_pbds::detail::left_child_next_sibling_↵
heap_const_iterator_< Node, _Alloc >::left_child_next_sibling_heap_const_iterator_ ( )
[inline]
```

Default constructor.

Definition at line 96 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

```
4.250.3.2 template<typename Node , typename _Alloc > __gnu_pbds::detail::left_child_next_sibling_↵
heap_const_iterator_< Node, _Alloc >::left_child_next_sibling_heap_const_iterator_ ( const
left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & other ) [inline]
```

Copy constructor.

Definition at line 101 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

#### 4.250.4 Member Function Documentation

```
4.250.4.1 template<typename Node , typename _Alloc > bool __gnu_pbds::detail::left_child_next_sibling_heap_↵
const_iterator_< Node, _Alloc >::operator!=( const left_child_next_sibling_heap_const_iterator_< Node,
_Alloc > & other ) const [inline]
```

Compares content (negatively) to a different iterator object.

Definition at line 111 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

```
4.250.4.2 template<typename Node , typename _Alloc > bool __gnu_pbds::detail::left_child_↵
next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator!=( const
left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > & other ) const [inline],
[inherited]
```

Compares content (negatively) to a different iterator object.

Definition at line 137 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

```
4.250.4.3 template<typename Node , typename _Alloc > const_reference __gnu_pbds::detail::left_child_next_↵
_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator*( ) const [inline],
[inherited]
```

Access.

Definition at line 124 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

```
4.250.4.4 template<typename Node , typename _Alloc > const_pointer __gnu_pbds::detail::left_child_next_↵
_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator-> ( ) const [inline],
[inherited]
```

Access.

Definition at line 116 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

```
4.250.4.5  template<typename Node , typename _Alloc > bool __gnu_pbds::detail::left_child_next_sibling_heap_↵
const_iterator_< Node, _Alloc >::operator==( const left_child_next_sibling_heap_const_iterator_< Node,
_Alloc > & other ) const    [inline]
```

Compares content to a different iterator object.

Definition at line 106 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

```
4.250.4.6  template<typename Node , typename _Alloc > bool __gnu_pbds::detail::left_child_↵
next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator==( const
left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > & other ) const    [inline],
[inherited]
```

Compares content to a different iterator object.

Definition at line 132 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

The documentation for this class was generated from the following file:

- [left\\_child\\_next\\_sibling\\_heap\\_/const\\_iterator.hpp](#)

#### 4.251 \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_< \_Value, \_Metadata, \_Alloc > Struct Template Reference

##### Public Types

- typedef \_Metadata **metadata\_type**
- typedef \_Alloc::template rebind< [this\\_type](#) >::other::pointer **node\_pointer**
- typedef \_Alloc::size\_type **size\_type**
- typedef \_Value **value\_type**

##### Public Attributes

- metadata\_type **m\_metadata**
- node\_pointer **m\_p\_l\_child**
- node\_pointer **m\_p\_next\_sibling**
- node\_pointer **m\_p\_prev\_or\_parent**
- value\_type **m\_value**

##### 4.251.1 Detailed Description

```
template<typename _Value, typename _Metadata, typename _Alloc> struct __gnu_pbds::detail::left_child_next_sibling_heap_node_↵
_< _Value, _Metadata, _Alloc >
```

Node.

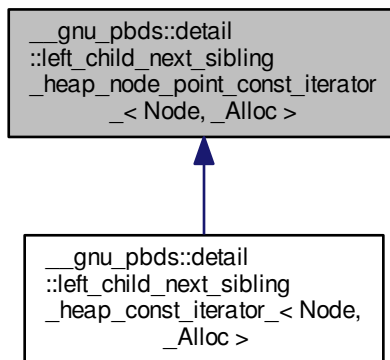
Definition at line 50 of file `left_child_next_sibling_heap_/node.hpp`.

The documentation for this struct was generated from the following file:

- [left\\_child\\_next\\_sibling\\_heap\\_/node.hpp](#)

## 4.252 `__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >`:



### Public Types

- `typedef _Alloc::template rebind< value_type >::other::const_pointer const_pointer`
- `typedef _Alloc::template rebind< value_type >::other::const_reference const_reference`
- `typedef trivial_iterator_difference_type difference_type`
- `typedef trivial_iterator_tag iterator_category`
- `typedef _Alloc::template rebind< value_type >::other::pointer pointer`
- `typedef _Alloc::template rebind< value_type >::other::reference reference`
- `typedef Node::value_type value_type`

### Public Member Functions

- `left_child_next_sibling_heap_node_point_const_iterator_ (node_pointer p_nd)`
- `left_child_next_sibling_heap_node_point_const_iterator_ ()`
- `left_child_next_sibling_heap_node_point_const_iterator_ (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other)`
- `bool operator!= (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other) const`
- `const_reference operator* () const`
- `const_pointer operator-> () const`
- `bool operator== (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other) const`

### Public Attributes

- node\_pointer `m_p_nd`

## Protected Types

- `typedef _Alloc::template rebind< Node >::other::pointer node_pointer`

### 4.252.1 Detailed Description

```
template<typename Node, typename _Alloc>class __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_↵
< Node, _Alloc >
```

Const point-type iterator.

Definition at line 61 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

### 4.252.2 Member Typedef Documentation

```
4.252.2.1 template<typename Node , typename _Alloc > typedef _Alloc::template rebind< value_type>::other::const_pointer
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::const_pointer
```

Iterator's const pointer type.

Definition at line 86 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

```
4.252.2.2 template<typename Node , typename _Alloc > typedef _Alloc::template rebind< value_type>::other::const_reference
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::const_reference
```

Iterator's const reference type.

Definition at line 98 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

```
4.252.2.3 template<typename Node , typename _Alloc > typedef trivial_iterator_difference_type
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::difference_type
```

Difference type.

Definition at line 71 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

```
4.252.2.4 template<typename Node , typename _Alloc > typedef trivial_iterator_tag __gnu_pbds_↵
::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::iterator_category
```

Category.

Definition at line 68 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

```
4.252.2.5 template<typename Node , typename _Alloc > typedef _Alloc::template rebind< value_type>::other::pointer
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::pointer
```

Iterator's pointer type.

Definition at line 80 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

4.252.2.6 `template<typename Node , typename _Alloc > typedef _Alloc::template rebind< value_type >::other::reference  
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc  
>::reference`

Iterator's reference type.

Definition at line 92 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

4.252.2.7 `template<typename Node , typename _Alloc > typedef Node::value_type __gnu_pbds::  
::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc  
>::value_type`

Iterator's value type.

Definition at line 74 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

#### 4.252.3 Constructor & Destructor Documentation

4.252.3.1 `template<typename Node , typename _Alloc > __gnu_pbds::detail::left_child_next_sibling_heap_node_↵  
point_const_iterator_< Node, _Alloc >::left_child_next_sibling_heap_node_point_const_iterator_( )  
[inline]`

Default constructor.

Definition at line 106 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

4.252.3.2 `template<typename Node , typename _Alloc > __gnu_pbds::detail::left_child_next_sibling_heap_node_↵  
point_const_iterator_< Node, _Alloc >::left_child_next_sibling_heap_node_point_const_iterator_(  
const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > & other ) [inline]`

Copy constructor.

Definition at line 111 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

#### 4.252.4 Member Function Documentation

4.252.4.1 `template<typename Node , typename _Alloc > bool __gnu_pbds::detail::left_child_↵  
next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator!= ( const  
left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > & other ) const [inline]`

Compares content (negatively) to a different iterator object.

Definition at line 137 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

4.252.4.2 `template<typename Node , typename _Alloc > const_reference __gnu_pbds::detail::left_↵  
child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator*( ) const  
[inline]`

Access.

Definition at line 124 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

4.252.4.3 `template<typename Node , typename _Alloc > const_pointer __gnu_pbds::detail::left_child_↵  
_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator-> ( ) const  
[inline]`

Access.

Definition at line 116 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

```
4.252.4.4  template<typename Node , typename _Alloc > bool __gnu_pbds::detail::left_child_↵
            next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator==( const
            left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > & other ) const  [inline]
```

Compares content to a different iterator object.

Definition at line 132 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

The documentation for this class was generated from the following file:

- [left\\_child\\_next\\_sibling\\_heap\\_/point\\_const\\_iterator.hpp](#)

## 4.253 \_\_gnu\_pbds::detail::lu\_counter\_metadata< Size\_Type > Class Template Reference

### Public Types

- typedef Size\_Type **size\_type**

### Friends

- class **lu\_counter\_policy\_base**< **size\_type** >

### 4.253.1 Detailed Description

```
template<typename Size_Type = std::size_t>class __gnu_pbds::detail::lu_counter_metadata< Size_Type >
```

A list-update metadata type that moves elements to the front of the list based on the counter algorithm.

Definition at line 51 of file `lu_counter_metadata.hpp`.

The documentation for this class was generated from the following file:

- [lu\\_counter\\_metadata.hpp](#)

## 4.254 \_\_gnu\_pbds::detail::lu\_counter\_policy\_base< Size\_Type > Class Template Reference

### Protected Types

- typedef Size\_Type **size\_type**

### Protected Member Functions

- [lu\\_counter\\_metadata](#)< size\_type > **operator()** (size\_type max\_size) const
- template<typename Metadata\_Reference > bool **operator()** (Metadata\_Reference r\_data, size\_type m\_max\_count) const

### 4.254.1 Detailed Description

#### 4.255 `__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >` Class Template Reference 767

```
template<typename Size_Type>class __gnu_pbds::detail::lu_counter_policy_base< Size_Type >
```

Base class for list-update counter policy.

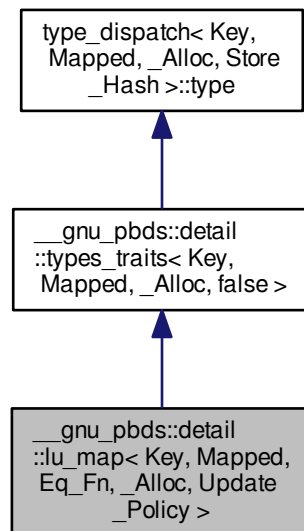
Definition at line 46 of file `lu_counter_metadata.hpp`.

The documentation for this class was generated from the following file:

- [lu\\_counter\\_metadata.hpp](#)

#### 4.255 `__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >`:



##### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `const_iterator` **const\_iterator**
- typedef `traits_base::const_pointer` **const\_pointer**
- typedef `traits_base::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `Eq_Fn` **eq\_fn**
- typedef `iterator` **iterator**
- typedef `traits_base::key_const_pointer` **key\_const\_pointer**
- typedef `traits_base::key_const_reference` **key\_const\_reference**
- typedef `traits_base::key_pointer` **key\_pointer**



- typedef traits\_base::key\_reference **key\_reference**
- typedef traits\_base::key\_type **key\_type**
- typedef traits\_base::mapped\_const\_pointer **mapped\_const\_pointer**
- typedef traits\_base::mapped\_const\_reference **mapped\_const\_reference**
- typedef traits\_base::mapped\_pointer **mapped\_pointer**
- typedef traits\_base::mapped\_reference **mapped\_reference**
- typedef traits\_base::mapped\_type **mapped\_type**
- typedef \_\_nothrowcopy::indicator **no\_throw\_indicator**
- typedef [point\\_const\\_iterator](#) **point\_const\_iterator**
- typedef [point\\_iterator](#) **point\_iterator**
- typedef traits\_base::pointer **pointer**
- typedef traits\_base::reference **reference**
- typedef \_Alloc::size\_type **size\_type**
- typedef integral\_constant< int, Store\_Hash > **store\_extra**
- typedef Update\_Policy::metadata\_type **update\_metadata**
- typedef Update\_Policy **update\_policy**
- typedef traits\_base::value\_type **value\_type**

#### Public Member Functions

- **lu\_map** (const [lu\\_map](#)< Key, Mapped, Eq\_Fn, \_Alloc, Update\_Policy > &)
- template<typename It > **lu\_map** (It, It)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- bool **erase** (key\_const\_reference)
- template<typename Pred > size\_type **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference r\_key)
- point\_const\_iterator **find** (key\_const\_reference r\_key) const
- [std::pair](#)< point\_iterator, bool > **insert** (const\_reference)
- size\_type **max\_size** () const
- mapped\_reference **operator[]** (key\_const\_reference r\_key)
- size\_type **size** () const
- void **swap** ([lu\\_map](#)< Key, Mapped, Eq\_Fn, \_Alloc, Update\_Policy > &)

#### Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

#### Protected Member Functions

- template<typename It > void **copy\_from\_range** (It, It)

## Friends

- class `const_iterator_`
- class `iterator_`

## 4.255.1 Detailed Description

template<typename Key, typename Mapped, typename Eq\_Fn, typename \_Alloc, typename Update\_Policy>class `__gnu_pbds::detail::lu_map`< Key, Mapped, Eq\_Fn, \_Alloc, Update\_Policy >

list-based (with updates) associative container. Skip to the lu, my darling.

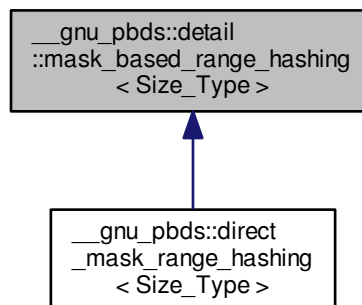
Definition at line 91 of file `lu_map.hpp`.

The documentation for this class was generated from the following file:

- [lu\\_map.hpp](#)

4.256 `__gnu_pbds::detail::mask_based_range_hashing< Size_Type >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::mask_based_range_hashing< Size_Type >`:



## Protected Types

- typedef `Size_Type` **size\_type**

## Protected Member Functions

- void **notify\_resized** (`size_type` size)
- `size_type` **range\_hash** (`size_type` hash) const
- void **swap** ([mask\\_based\\_range\\_hashing](#) &other)

#### 4.256.1 Detailed Description

```
template<typename Size_Type>class __gnu_pbds::detail::mask_based_range_hashing< Size_Type >
```

Range hashing policy.

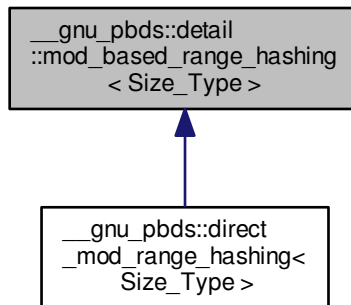
Definition at line 50 of file mask\_based\_range\_hashing.hpp.

The documentation for this class was generated from the following file:

- [mask\\_based\\_range\\_hashing.hpp](#)

#### 4.257 \_\_gnu\_pbds::detail::mod\_based\_range\_hashing< Size\_Type > Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::mod\_based\_range\_hashing< Size\_Type >:



##### Protected Types

- typedef Size\_Type **size\_type**

##### Protected Member Functions

- void **notify\_resized** (size\_type s)
- size\_type **range\_hash** (size\_type s) const
- void **swap** ([mod\\_based\\_range\\_hashing](#) &other)

#### 4.257.1 Detailed Description

```
template<typename Size_Type>class __gnu_pbds::detail::mod_based_range_hashing< Size_Type >
```

Mod based range hashing.

Definition at line 50 of file mod\_based\_range\_hashing.hpp.

The documentation for this class was generated from the following file:

- [mod\\_based\\_range\\_hashing.hpp](#)

## 4.258 `__gnu_pbds::detail::no_throw_copies< Key, Mapped >` Struct Template Reference

### Public Types

- `typedef integral_constant< int, __simple > indicator`

### Static Public Attributes

- `static const bool __simple`

#### 4.258.1 Detailed Description

`template<typename Key, typename Mapped>struct __gnu_pbds::detail::no_throw_copies< Key, Mapped >`

Primary template.

Definition at line 61 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

## 4.259 `__gnu_pbds::detail::no_throw_copies< Key, null_type >` Struct Template Reference

### Public Types

- `typedef integral_constant< int, is_simple< Key >::value > indicator`

#### 4.259.1 Detailed Description

`template<typename Key>struct __gnu_pbds::detail::no_throw_copies< Key, null_type >`

Specialization.

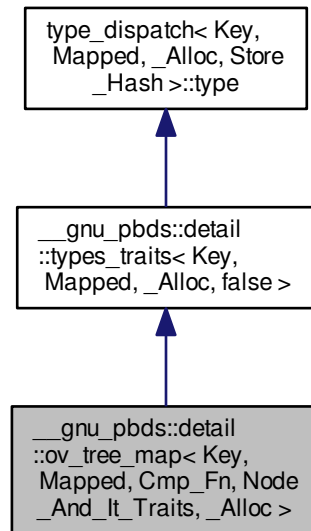
Definition at line 70 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

#### 4.260 `__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`:



#### Classes

- class `cond_dtor`

#### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `point_const_iterator` **const\_iterator**
- typedef `traits_base::const_pointer` **const\_pointer**
- typedef `traits_base::const_reference` **const\_reference**
- typedef `ov_tree_tag` **container\_category**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `point_iterator` **iterator**
- typedef `traits_base::key_const_pointer` **key\_const\_pointer**
- typedef `traits_base::key_const_reference` **key\_const\_reference**
- typedef `traits_base::key_pointer` **key\_pointer**
- typedef `traits_base::key_reference` **key\_reference**
- typedef `traits_base::key_type` **key\_type**
- typedef `traits_base::mapped_const_pointer` **mapped\_const\_pointer**

- `typedef traits_base::mapped_const_reference mapped_const_reference`
- `typedef traits_base::mapped_pointer mapped_pointer`
- `typedef traits_base::mapped_reference mapped_reference`
- `typedef traits_base::mapped_type mapped_type`
- `typedef __nothrowcopy::indicator no_throw_indicator`
- `typedef traits_type::node_const_iterator node_const_iterator`
- `typedef traits_type::node_iterator node_iterator`
- `typedef traits_type::node_update node_update`
- `typedef const_pointer point_const_iterator`
- `typedef pointer point_iterator`
- `typedef traits_base::pointer pointer`
- `typedef traits_base::reference reference`
- `typedef _Alloc::size_type size_type`
- `typedef integral_constant< int, Store_Hash > store_extra`
- `typedef traits_base::value_type value_type`

#### Public Member Functions

- `ov_tree_map` (const Cmp\_Fn &)
- `ov_tree_map` (const Cmp\_Fn &, const node\_update &)
- `ov_tree_map` (const `ov_tree_map`< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- iterator `begin` ()
- const\_iterator `begin` () const
- void `clear` ()
- `template<typename It > void copy_from_range` (It, It)
- bool `empty` () const
- iterator `end` ()
- const\_iterator `end` () const
- bool `erase` (key\_const\_reference)
- iterator `erase` (iterator it)
- `template<typename Pred > size_type erase_if` (Pred)
- point\_iterator `find` (key\_const\_reference r\_key)
- point\_const\_iterator `find` (key\_const\_reference r\_key) const
- Cmp\_Fn & `get_cmp_fn` ()
- const Cmp\_Fn & `get_cmp_fn` () const
- `std::pair`< point\_iterator, bool > `insert` (const\_reference r\_value)
- void `join` (`ov_tree_map`< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator `lower_bound` (key\_const\_reference r\_key)
- point\_const\_iterator `lower_bound` (key\_const\_reference r\_key) const
- size\_type `max_size` () const
- node\_const\_iterator `node_begin` () const
- node\_iterator `node_begin` ()
- node\_const\_iterator `node_end` () const
- node\_iterator `node_end` ()
- mapped\_reference `operator[]` (key\_const\_reference r\_key)
- size\_type `size` () const
- void `split` (key\_const\_reference, `ov_tree_map`< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void `swap` (`ov_tree_map`< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator `upper_bound` (key\_const\_reference r\_key)
- point\_const\_iterator `upper_bound` (key\_const\_reference r\_key) const

## Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

## 4.260.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >
```

Ordered-vector tree associative-container.

Definition at line 106 of file ov\_tree\_map\_.hpp.

## 4.260.2 Member Function Documentation

```
4.260.2.1 template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
        _Alloc > ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator
        __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin ( ) const
        [inline]
```

Returns a const node\_iterator corresponding to the node at the root of the tree.

Definition at line 45 of file ov\_tree\_map\_.hpp.

```
4.260.2.2 template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits ,
        typename _Alloc > ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator
        __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin ( )
        [inline]
```

Returns a node\_iterator corresponding to the node at the root of the tree.

Definition at line 57 of file ov\_tree\_map\_.hpp.

```
4.260.2.3 template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
        _Alloc > ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator
        __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ( ) const
        [inline]
```

Returns a const node\_iterator corresponding to a node just after a leaf of the tree.

Definition at line 51 of file ov\_tree\_map\_.hpp.

```
4.260.2.4 template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits ,
        typename _Alloc > ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator
        __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ( )
        [inline]
```

Returns a node\_iterator corresponding to a node just after a leaf of the tree.

Definition at line 63 of file ov\_tree\_map\_.hpp.

The documentation for this class was generated from the following file:

- [ov\\_tree\\_map\\_.hpp](#)

## 4.261 `__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type >` Class Template Reference

### Public Member Functions

- **cond\_dtor** (value\_vector a\_vec, iterator &r\_last\_it, Size\_Type total\_size)
- void **set\_no\_action** ()

### Protected Attributes

- value\_vector **m\_a\_vec**
- const Size\_Type **m\_max\_size**
- bool **m\_no\_action**
- iterator & **m\_r\_last\_it**

#### 4.261.1 Detailed Description

template<typename Key, typename Mapped, typename Cmp\_Fn, typename Node\_And\_It\_Traits, typename \_Alloc>template<typename Size\_Type>class `__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type >`

Conditional destructor.

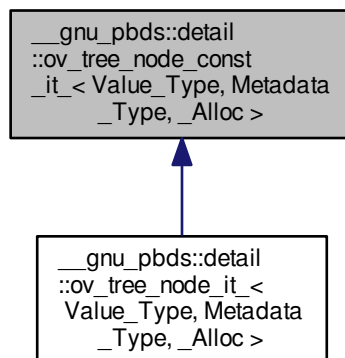
Definition at line 182 of file `ov_tree_map_.hpp`.

The documentation for this class was generated from the following file:

- [ov\\_tree\\_map\\_.hpp](#)

## 4.262 `__gnu_pbds::detail::ov_tree_node_const_it< Value_Type, Metadata_Type, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::ov_tree_node_const_it< Value_Type, Metadata_Type, _Alloc >`:





## Public Types

- typedef `_Alloc::template rebind< typename remove_const< Value_Type >::type >::other::const_pointer` **const\_reference**
- typedef `trivial_iterator_difference_type` **difference\_type**
- typedef `trivial_iterator_tag` **iterator\_category**
- typedef `_Alloc::template rebind< metadata_type >::other::const_reference` **metadata\_const\_reference**
- typedef `Metadata_Type` **metadata\_type**
- typedef `_Alloc::template rebind< typename remove_const< Value_Type >::type >::other::const_pointer` **reference**
- typedef `_Alloc::template rebind< Value_Type >::other::const_pointer` **value\_type**

## Public Member Functions

- **ov\_tree\_node\_const\_it\_** (const\_pointer p\_nd=0, const\_pointer p\_begin\_nd=0, const\_pointer p\_end\_nd=0, const\_metadata\_pointer p\_metadata=0)
- `this_type get_l_child` () const
- `metadata_const_reference get_metadata` () const
- `this_type get_r_child` () const
- bool **operator!=** (const `this_type` &other) const
- const\_reference **operator\*** () const
- bool **operator==** (const `this_type` &other) const

## Public Attributes

- pointer **m\_p\_begin\_value**
- pointer **m\_p\_end\_value**
- const\_metadata\_pointer **m\_p\_metadata**
- pointer **m\_p\_value**

## Protected Types

- typedef `_Alloc::template rebind< Metadata_Type >::other::const_pointer` **const\_metadata\_pointer**
- typedef `_Alloc::template rebind< Value_Type >::other::const_pointer` **const\_pointer**
- typedef `_Alloc::template rebind< Value_Type >::other::pointer` **pointer**
- typedef `ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >` **this\_type**

## Static Protected Member Functions

- `template<typename Ptr > static Ptr` **mid\_pointer** (Ptr p\_begin, Ptr p\_end)

## 4.262.1 Detailed Description

`template<typename Value_Type, typename Metadata_Type, typename _Alloc> class __gnu_pbds::detail::ov_tree_node_const_it_` `< Value_Type, Metadata_Type, _Alloc >`

Const node reference.

Definition at line 57 of file `ov_tree_map_/node_iterators.hpp`.

#### 4.262.2 Member Function Documentation

4.262.2.1 `template<typename Value_Type , typename Metadata_Type , typename _Alloc > this_type  
__gnu_pbds::detail::ov_tree_node_const_it< Value_Type, Metadata_Type, _Alloc >::get_l_child ( ) const`  
[inline]

Returns the node iterator associated with the left node.

Definition at line 142 of file `ov_tree_map_/node_iterators.hpp`.

4.262.2.2 `template<typename Value_Type , typename Metadata_Type , typename _Alloc > this_type  
__gnu_pbds::detail::ov_tree_node_const_it< Value_Type, Metadata_Type, _Alloc >::get_r_child ( ) const`  
[inline]

Returns the node iterator associated with the right node.

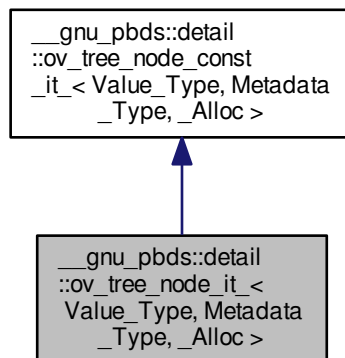
Definition at line 158 of file `ov_tree_map_/node_iterators.hpp`.

The documentation for this class was generated from the following file:

- [ov\\_tree\\_map\\_/node\\_iterators.hpp](#)

#### 4.263 `__gnu_pbds::detail::ov_tree_node_it< Value_Type, Metadata_Type, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::ov_tree_node_it< Value_Type, Metadata_Type, _Alloc >`:



#### Public Types

- `typedef _Alloc::template rebind< typename remove_const< Value_Type >::type >::other::pointer const_↔reference`
- `typedef trivial\_iterator\_difference\_type difference_type`
- `typedef trivial\_iterator\_tag iterator_category`
- `typedef _Alloc::template rebind< metadata_type >::other::const_reference metadata_const_reference`

- typedef Metadata\_Type **metadata\_type**
- typedef \_Alloc::template rebind< typename remove\_const< Value\_Type >::type >::other::pointer **reference**
- typedef \_Alloc::template rebind< Value\_Type >::other::pointer **value\_type**

#### Public Member Functions

- **ov\_tree\_node\_it\_** (const\_pointer p\_nd=0, const\_pointer p\_begin\_nd=0, const\_pointer p\_end\_nd=0, const\_pointer metadata\_pointer p\_metadata=0)
- **ov\_tree\_node\_it\_** **get\_l\_child** () const
- metadata\_const\_reference **get\_metadata** () const
- **ov\_tree\_node\_it\_** **get\_r\_child** () const
- bool **operator!=** (const **this\_type** &other) const
- reference **operator\*** () const
- bool **operator==** (const **this\_type** &other) const

#### Public Attributes

- pointer **m\_p\_begin\_value**
- pointer **m\_p\_end\_value**
- const\_metadata\_pointer **m\_p\_metadata**
- pointer **m\_p\_value**

#### Static Protected Member Functions

- template<typename Ptr > static Ptr **mid\_pointer** (Ptr p\_begin, Ptr p\_end)

#### 4.263.1 Detailed Description

template<typename Value\_Type, typename Metadata\_Type, typename \_Alloc>class \_\_gnu\_pbds::detail::ov\_tree\_node\_it\_ < Value\_Type, Metadata\_Type, \_Alloc >

Node reference.

Definition at line 204 of file ov\_tree\_map\_/node\_iterators.hpp.

#### 4.263.2 Member Function Documentation

4.263.2.1 template<typename Value\_Type , typename Metadata\_Type , typename \_Alloc > **ov\_tree\_node\_it\_** \_\_gnu\_pbds::detail::ov\_tree\_node\_it\_ < Value\_Type, Metadata\_Type, \_Alloc >::get\_l\_child ( ) const  
[inline]

Returns the node reference associated with the left node.

Definition at line 252 of file ov\_tree\_map\_/node\_iterators.hpp.

4.263.2.2 template<typename Value\_Type , typename Metadata\_Type , typename \_Alloc > **ov\_tree\_node\_it\_** \_\_gnu\_pbds::detail::ov\_tree\_node\_it\_ < Value\_Type, Metadata\_Type, \_Alloc >::get\_r\_child ( ) const  
[inline]

Returns the node reference associated with the right node.

Definition at line 268 of file ov\_tree\_map\_/node\_iterators.hpp.

4.263.2.3 `template<typename Value_Type , typename Metadata_Type , typename _Alloc > reference  
 __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::operator* ( ) const  
 [inline]`

Access.

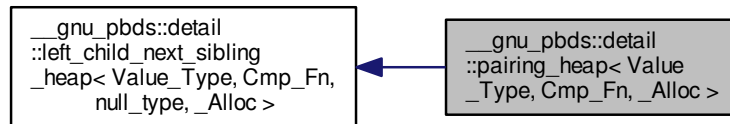
Definition at line 247 of file `ov_tree_map_/node_iterators.hpp`.

The documentation for this class was generated from the following file:

- [ov\\_tree\\_map\\_/node\\_iterators.hpp](#)

## 4.264 `__gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >`:



### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `__rebind_a::const_pointer` **const\_pointer**
- typedef `__rebind_a::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::point_const_iterator` **point\_const\_iterator**
- typedef `base_type::point_iterator` **point\_iterator**
- typedef `__rebind_a::pointer` **pointer**
- typedef `__rebind_a::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `Value_Type` **value\_type**

### Public Member Functions

- **pairing\_heap** (`const Cmp_Fn &`)
- **pairing\_heap** (`const pairing\_heap &`)
- **iterator begin** ()
- **const\_iterator begin** () `const`
- `void clear` ()
- `bool empty` () `const`

- `iterator end ()`
- `const_iterator end () const`
- `void erase (point_iterator)`
- `template<typename Pred > size_type erase_if (Pred)`
- `Cmp_Fn & get_cmp_fn ()`
- `const Cmp_Fn & get_cmp_fn () const`
- `void join (pairing_heap &)`
- `size_type max_size () const`
- `void modify (point_iterator, const_reference)`
- `void pop ()`
- `point_iterator push (const_reference)`
- `size_type size () const`
- `template<typename Pred > void split (Pred, pairing_heap &)`
- `void swap (pairing_heap &)`
- `void swap (left_child_next_sibling_heap< Value_Type, Cmp_Fn, null_type, _Alloc > &)`
- `const_reference top () const`

### Protected Types

- `typedef node_allocator::value_type node`
- `typedef _Alloc::template rebind< left_child_next_sibling_heap_node_< Value_Type, null_type, _Alloc > >::other node_allocator`
- `typedef node_allocator::const_pointer node_const_pointer`
- `typedef null_type node_metadata`
- `typedef std::pair< node_pointer, node_pointer > node_pointer_pair`

### Protected Member Functions

- `void actual_erase_node (node_pointer)`
- `void bubble_to_top (node_pointer)`
- `void clear_imp (node_pointer)`
- `template<typename It > void copy_from_range (It, It)`
- `node_pointer get_new_node_for_insert (const_reference)`
- `node_pointer prune (Pred)`
- `void swap_with_parent (node_pointer, node_pointer)`
- `void to_linked_list ()`
- `void value_swap (left_child_next_sibling_heap &)`

### Static Protected Member Functions

- `static void make_child_of (node_pointer, node_pointer)`
- `static node_pointer parent (node_pointer)`

### Protected Attributes

- `node_pointer m_p_root`
- `size_type m_size`

## 4.264.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>class __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >
```

Pairing heap.

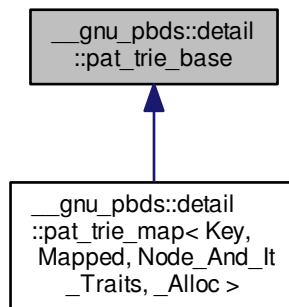
Definition at line 77 of file pairing\_heap\_.hpp.

The documentation for this class was generated from the following file:

- [pairing\\_heap\\_.hpp](#)

## 4.265 \_\_gnu\_pbds::detail::pat\_trie\_base Struct Reference

Inheritance diagram for \_\_gnu\_pbds::detail::pat\_trie\_base:



## Classes

- class [\\_CIter](#)
- struct [\\_Head](#)
- struct [\\_Inode](#)
- class [\\_Iter](#)
- struct [\\_Leaf](#)
- struct [\\_Metadata](#)
- struct [\\_Metadata< null\\_type, \\_Alloc >](#)
- struct [\\_Node\\_base](#)
- class [\\_Node\\_citer](#)
- class [\\_Node\\_iter](#)

## Public Types

- enum [node\\_type](#) { [i\\_node](#), [leaf\\_node](#), [head\\_node](#) }

#### 4.265.1 Detailed Description

Base type for PATRICIA trees.

Definition at line 51 of file pat\_trie\_base.hpp.

#### 4.265.2 Member Enumeration Documentation

##### 4.265.2.1 enum \_\_gnu\_pbds::detail::pat\_trie\_base::node\_type

Three types of nodes.

i\_node is used by \_Inode, leaf\_node by \_Leaf, and head\_node by \_Head.

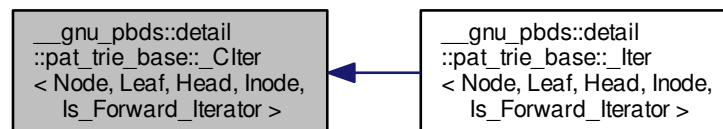
Definition at line 58 of file pat\_trie\_base.hpp.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

#### 4.266 \_\_gnu\_pbds::detail::pat\_trie\_base::\_Clter< Node, Leaf, Head, Inode, Is\_Forward\_Iterator > Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::pat\_trie\_base::\_Clter< Node, Leaf, Head, Inode, Is\_Forward\_Iterator >:



#### Public Types

- typedef \_Alloc::template rebind< Head > **\_\_rebind\_h**
- typedef \_Alloc::template rebind< Inode > **\_\_rebind\_in**
- typedef \_Alloc::template rebind< Leaf > **\_\_rebind\_l**
- typedef \_Alloc::template rebind< Node > **\_\_rebind\_n**
- typedef allocator\_type **\_Alloc**
- typedef Node::allocator\_type **allocator\_type**
- typedef type\_traits::const\_pointer **const\_pointer**
- typedef type\_traits::const\_reference **const\_reference**
- typedef allocator\_type::difference\_type **difference\_type**
- typedef \_\_rebind\_h::other::pointer **head\_pointer**
- typedef Inode::iterator **inode\_iterator**
- typedef \_\_rebind\_in::other::pointer **inode\_pointer**
- typedef [std::bidirectional\\_iterator\\_tag](#) **iterator\_category**

- typedef `__rebind_l::other::const_pointer` **leaf\_const\_pointer**
- typedef `__rebind_l::other::pointer` **leaf\_pointer**
- typedef `__rebind_n::other::pointer` **node\_pointer**
- typedef `type_traits::pointer` **pointer**
- typedef `type_traits::reference` **reference**
- typedef `Node::type_traits` **type\_traits**
- typedef `type_traits::value_type` **value\_type**

#### Public Member Functions

- `_Clter` (`node_pointer p_nd=0`)
- `_Clter` (`const _Clter< Node, Leaf, Head, Inode,!Is_Forward_Iterator > &other`)
- `bool operator!=` (`const _Clter &other`) `const`
- `bool operator!=` (`const _Clter< Node, Leaf, Head, Inode,!Is_Forward_Iterator > &other`) `const`
- `const_reference operator*` (`()`) `const`
- `_Clter & operator++` (`()`)
- `_Clter operator++` (`int`)
- `_Clter & operator--` (`()`)
- `_Clter operator--` (`int`)
- `const_pointer operator->` (`()`) `const`
- `_Clter & operator=` (`const _Clter &other`)
- `_Clter & operator=` (`const _Clter< Node, Leaf, Head, Inode,!Is_Forward_Iterator > &other`)
- `bool operator==` (`const _Clter &other`) `const`
- `bool operator==` (`const _Clter< Node, Leaf, Head, Inode,!Is_Forward_Iterator > &other`) `const`

#### Public Attributes

- `node_pointer` **m\_p\_nd**

#### Protected Member Functions

- `void dec` (`false_type`)
- `void dec` (`true_type`)
- `void inc` (`false_type`)
- `void inc` (`true_type`)

#### Static Protected Member Functions

- `static node_pointer get_larger_sibling` (`node_pointer p_nd`)
- `static node_pointer get_smaller_sibling` (`node_pointer p_nd`)
- `static leaf_pointer leftmost_descendant` (`node_pointer p_nd`)
- `static leaf_pointer rightmost_descendant` (`node_pointer p_nd`)



## 4.266.1 Detailed Description

```
template<typename Node, typename Leaf, typename Head, typename Inode, bool Is_Forward_Iterator>class __gnu_pbds::detail<
::pat_trie_base::_Clter< Node, Leaf, Head, Inode, Is_Forward_Iterator >
```

Const iterator.

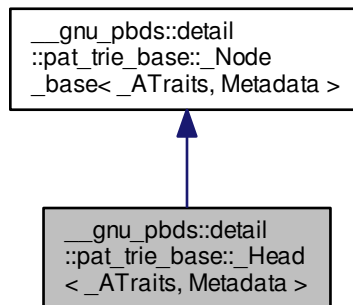
Definition at line 487 of file pat\_trie\_base.hpp.

The documentation for this class was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 4.267 \_\_gnu\_pbds::detail::pat\_trie\_base::\_Head&lt; \_ATraits, Metadata &gt; Struct Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::pat\_trie\_base::\_Head< \_ATraits, Metadata >:



## Public Types

- typedef \_Alloc::template rebind< \_ATraits > **\_\_rebind\_at**
- typedef \_Alloc::template rebind< [\\_Node\\_base](#) > **\_\_rebind\_n**
- typedef \_ATraits::const\_iterator **a\_const\_iterator**
- typedef \_\_rebind\_at::other::const\_pointer **a\_const\_pointer**
- typedef \_ATraits **access\_traits**
- typedef \_Alloc **allocator\_type**
- typedef [\\_Node\\_base](#)< \_ATraits, Metadata > **base\_type**
- typedef base\_type::node\_pointer **node\_pointer**
- typedef base\_type::type\_traits **type\_traits**

## Public Attributes

- node\_pointer **m\_p\_max**
- node\_pointer **m\_p\_min**
- node\_pointer **m\_p\_parent**
- const [node\\_type](#) **m\_type**

## 4.267.1 Detailed Description

```
template<typename _ATraits, typename Metadata> struct __gnu_pbds::detail::pat_trie_base::_Head<_ATraits, Metadata>
```

Head node for PATRICIA tree.

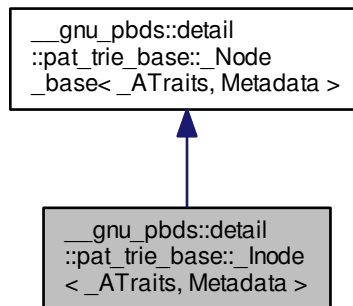
Definition at line 131 of file `pat_trie_base.hpp`.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

4.268 `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>`:



## Classes

- struct [const\\_iterator](#)
- struct [iterator](#)

## Public Types

- enum { **arr\_size** }
- typedef `_Alloc::template rebind<_ATraits>` **\_\_rebind\_at**
- typedef `_Alloc::template rebind<node_pointer>::other` **\_\_rebind\_np**
- typedef `base_type::allocator_type` **\_Alloc**
- typedef `base_type::access_traits` **access\_traits**
- typedef `_Alloc` **allocator\_type**
- typedef `_Node_base<_ATraits, Metadata>` **base\_type**
- typedef `__rebind_np::pointer` **node\_pointer\_pointer**
- typedef `__rebind_np::reference` **node\_pointer\_reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `base_type::type_traits` **type\_traits**
- typedef `type_traits::value_type` **value\_type**

## Public Member Functions

- **\_Inode** (size\_type, const a\_const\_iterator)
- node\_pointer **add\_child** (node\_pointer, a\_const\_iterator, a\_const\_iterator, a\_const\_pointer)
- [const\\_iterator](#) **begin** () const
- [iterator](#) **begin** ()
- [const\\_iterator](#) **end** () const
- [iterator](#) **end** ()
- [iterator](#) **get\_child\_it** (a\_const\_iterator, a\_const\_iterator, a\_const\_pointer)
- node\_pointer **get\_child\_node** (a\_const\_iterator, a\_const\_iterator, a\_const\_pointer)
- node\_const\_pointer **get\_child\_node** (a\_const\_iterator, a\_const\_iterator, a\_const\_pointer) const
- size\_type **get\_e\_ind** () const
- node\_const\_pointer **get\_join\_child** (node\_const\_pointer, a\_const\_pointer) const
- node\_pointer **get\_join\_child** (node\_pointer, a\_const\_pointer)
- node\_pointer **get\_lower\_bound\_child\_node** (a\_const\_iterator, a\_const\_iterator, size\_type, a\_const\_pointer)
- leaf\_pointer **leftmost\_descendant** ()
- leaf\_const\_pointer **leftmost\_descendant** () const
- a\_const\_iterator **pref\_b\_it** () const
- a\_const\_iterator **pref\_e\_it** () const
- void **remove\_child** (node\_pointer)
- void **remove\_child** ([iterator](#))
- void **replace\_child** (node\_pointer, a\_const\_iterator, a\_const\_iterator, a\_const\_pointer)
- leaf\_pointer **rightmost\_descendant** ()
- leaf\_const\_pointer **rightmost\_descendant** () const
- bool **should\_be\_mine** (a\_const\_iterator, a\_const\_iterator, size\_type, a\_const\_pointer) const
- void **update\_prefixes** (a\_const\_pointer)

## Public Attributes

- node\_pointer **m\_p\_parent**
- const [node\\_type](#) **m\_type**

## 4.268.1 Detailed Description

```
template<typename _ATraits, typename Metadata> struct __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >
```

Internal node type, PATRICIA tree.

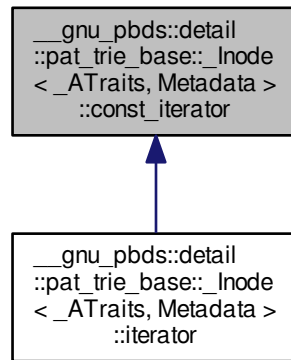
Definition at line 211 of file pat\_trie\_base.hpp.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

4.269 \_\_gnu\_pbds::detail::pat\_trie\_base::\_Inode<\_ATraits, Metadata>::const\_iterator Struct Reference

Inheritance diagram for \_\_gnu\_pbds::detail::pat\_trie\_base::\_Inode<\_ATraits, Metadata>::const\_iterator:



## Public Types

- typedef `_Alloc::difference_type` **difference\_type**
- typedef `std::forward_iterator_tag` **iterator\_category**
- typedef `node_pointer_pointer` **pointer**
- typedef `node_pointer_reference` **reference**
- typedef `node_pointer` **value\_type**

## Public Member Functions

- **const\_iterator** (`node_pointer_pointer p_p_cur=0, node_pointer_pointer p_p_end=0`)
- bool **operator!=** (const `const_iterator` &other) const
- `node_const_pointer` **operator\*** () const
- `const_iterator` & **operator++** ()
- `const_iterator` **operator++** (int)
- `const node_pointer_pointer` **operator->** () const
- bool **operator==** (const `const_iterator` &other) const

## Public Attributes

- `node_pointer_pointer` **m\_p\_p\_cur**
- `node_pointer_pointer` **m\_p\_p\_end**

## 4.269.1 Detailed Description

```
template<typename ATraits, typename Metadata>struct __gnu_pbds::detail::pat_trie_base::_Inode< ATraits, Metadata >::const_iterator
```

Constant child iterator.

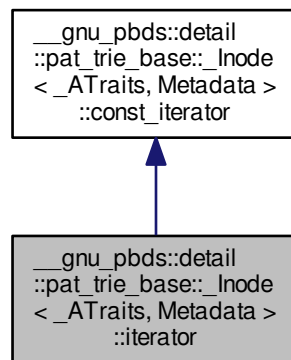
Definition at line 255 of file pat\_trie\_base.hpp.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 4.270 \_\_gnu\_pbds::detail::pat\_trie\_base::\_Inode&lt; ATraits, Metadata &gt;::iterator Struct Reference

Inheritance diagram for \_\_gnu\_pbds::detail::pat\_trie\_base::\_Inode< ATraits, Metadata >::iterator:



## Public Types

- typedef `_Alloc::difference_type` **difference\_type**
- typedef `std::forward_iterator_tag` **iterator\_category**
- typedef `node_pointer_pointer` **pointer**
- typedef `node_pointer_reference` **reference**
- typedef `node_pointer` **value\_type**

## Public Member Functions

- **iterator** (`node_pointer_pointer p_p_cur=0, node_pointer_pointer p_p_end=0`)
- bool **operator!=** (`const const_iterator &other`) const
- bool **operator!=** (`const iterator &other`) const
- `node_const_pointer` **operator\*** () const
- `node_pointer` **operator\*** ()

- [iterator](#) & **operator++** ()
- [iterator](#) **operator++** (int)
- const node\_pointer\_pointer **operator->** () const
- node\_pointer\_pointer **operator->** ()
- bool **operator==** (const [const\\_iterator](#) &other) const
- bool **operator==** (const [iterator](#) &other) const

#### Public Attributes

- node\_pointer\_pointer **m\_p\_p\_cur**
- node\_pointer\_pointer **m\_p\_p\_end**

#### 4.270.1 Detailed Description

template<typename ATraits, typename Metadata> struct `__gnu_pbds::detail::pat_trie_base::_Inode< ATraits, Metadata >::iterator`

Child iterator.

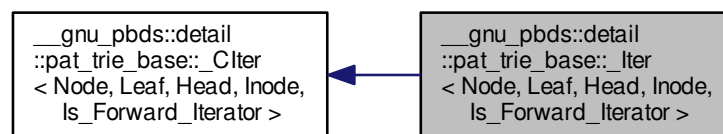
Definition at line 320 of file `pat_trie_base.hpp`.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

#### 4.271 `__gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >`:



#### Public Types

- typedef `_Alloc::template rebind< Head > __rebind_h`
- typedef `_Alloc::template rebind< Inode > __rebind_in`
- typedef `_Alloc::template rebind< Leaf > __rebind_l`
- typedef `_Alloc::template rebind< Node > __rebind_n`
- typedef `allocator_type _Alloc`
- typedef `base_type::allocator_type allocator_type`
- typedef `_CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator > base_type`

- typedef type\_traits::const\_pointer **const\_pointer**
- typedef type\_traits::const\_reference **const\_reference**
- typedef allocator\_type::difference\_type **difference\_type**
- typedef base\_type::head\_pointer **head\_pointer**
- typedef Inode::iterator **inode\_iterator**
- typedef base\_type::inode\_pointer **inode\_pointer**
- typedef [std::bidirectional\\_iterator\\_tag](#) **iterator\_category**
- typedef base\_type::leaf\_const\_pointer **leaf\_const\_pointer**
- typedef base\_type::leaf\_pointer **leaf\_pointer**
- typedef base\_type::node\_pointer **node\_pointer**
- typedef type\_traits::pointer **pointer**
- typedef type\_traits::reference **reference**
- typedef base\_type::type\_traits **type\_traits**
- typedef type\_traits::value\_type **value\_type**

#### Public Member Functions

- **\_Iter** (node\_pointer p\_nd=0)
- **\_Iter** (const **\_Iter**< Node, Leaf, Head, Inode,!Is\_Forward\_Iterator > &other)
- bool **operator!=** (const **\_CIter** &other) const
- bool **operator!=** (const **\_CIter**< Node, Leaf, Head, Inode,!Is\_Forward\_Iterator > &other) const
- reference **operator\*** () const
- **\_Iter** & **operator++** ()
- **\_Iter** **operator++** (int)
- **\_Iter** & **operator--** ()
- **\_Iter** **operator--** (int)
- pointer **operator->** () const
- **\_Iter** & **operator=** (const **\_Iter** &other)
- **\_Iter** & **operator=** (const **\_Iter**< Node, Leaf, Head, Inode,!Is\_Forward\_Iterator > &other)
- bool **operator==** (const **\_CIter** &other) const
- bool **operator==** (const **\_CIter**< Node, Leaf, Head, Inode,!Is\_Forward\_Iterator > &other) const

#### Public Attributes

- node\_pointer **m\_p\_nd**

#### Protected Member Functions

- void **dec** (false\_type)
- void **dec** (true\_type)
- void **inc** (false\_type)
- void **inc** (true\_type)

#### Static Protected Member Functions

- static node\_pointer **get\_larger\_sibling** (node\_pointer p\_nd)
- static node\_pointer **get\_smaller\_sibling** (node\_pointer p\_nd)
- static leaf\_pointer **leftmost\_descendant** (node\_pointer p\_nd)
- static leaf\_pointer **rightmost\_descendant** (node\_pointer p\_nd)

## 4.271.1 Detailed Description

```
template<typename Node, typename Leaf, typename Head, typename Inode, bool Is_Forward_Iterator>class __gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >
```

Iterator.

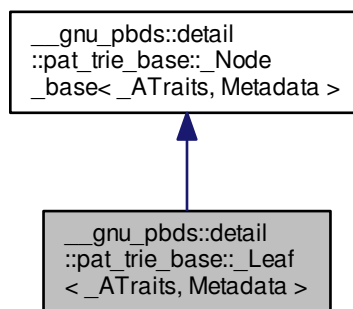
Definition at line 713 of file pat\_trie\_base.hpp.

The documentation for this class was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 4.272 \_\_gnu\_pbds::detail::pat\_trie\_base::\_Leaf&lt;\_ATraits, Metadata &gt; Struct Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::pat\_trie\_base::\_Leaf<\_ATraits, Metadata >:



## Public Types

- typedef \_Alloc::template rebind<\_ATraits > **\_\_rebind\_at**
- typedef \_Alloc::template rebind<\_Node\_base > **\_\_rebind\_n**
- typedef \_ATraits::const\_iterator **a\_const\_iterator**
- typedef \_\_rebind\_at::other::const\_pointer **a\_const\_pointer**
- typedef \_ATraits **access\_traits**
- typedef \_Alloc **allocator\_type**
- typedef \_Node\_base<\_ATraits, Metadata > **base\_type**
- typedef type\_traits::const\_reference **const\_reference**
- typedef \_\_rebind\_n::other::pointer **node\_pointer**
- typedef type\_traits::reference **reference**
- typedef base\_type::type\_traits **type\_traits**
- typedef type\_traits::value\_type **value\_type**



### Public Member Functions

- **\_Leaf** (const\_reference other)
- reference **value** ()
- const\_reference **value** () const

### Public Attributes

- node\_pointer **m\_p\_parent**
- const [node\\_type](#) **m\_type**

#### 4.272.1 Detailed Description

template<typename ATraits, typename Metadata> struct \_\_gnu\_pbds::detail::pat\_trie\_base::\_Leaf< ATraits, Metadata >

Leaf node for PATRICIA tree.

Definition at line 162 of file pat\_trie\_base.hpp.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

#### 4.273 \_\_gnu\_pbds::detail::pat\_trie\_base::\_Metadata< Metadata, \_Alloc > Struct Template Reference

### Public Types

- typedef \_Alloc::template rebind< Metadata > **\_\_rebind\_m**
- typedef \_Alloc **allocator\_type**
- typedef \_\_rebind\_m::other::const\_reference **const\_reference**
- typedef Metadata **metadata\_type**

### Public Member Functions

- const\_reference **get\_metadata** () const

### Public Attributes

- metadata\_type **m\_metadata**

#### 4.273.1 Detailed Description

template<typename Metadata, typename \_Alloc> struct \_\_gnu\_pbds::detail::pat\_trie\_base::\_Metadata< Metadata, \_Alloc >

Metadata base primary template.

Definition at line 67 of file pat\_trie\_base.hpp.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

4.274 `__gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc >` Struct Template Reference

## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `null_type` **metadata\_type**

## 4.274.1 Detailed Description

template<typename `_Alloc`>struct `__gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc >`

Specialization for null metadata.

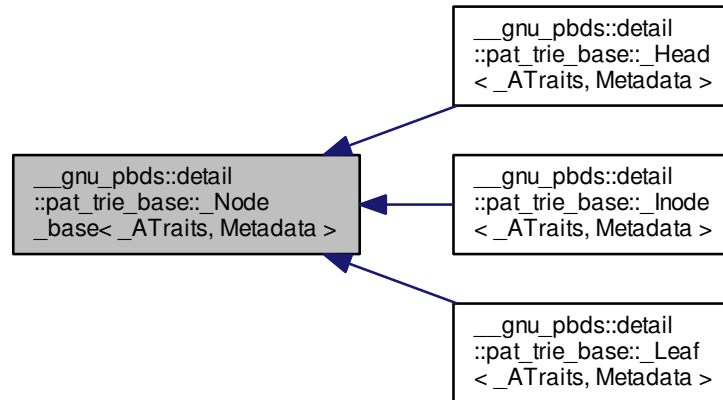
Definition at line 83 of file `pat_trie_base.hpp`.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

4.275 `__gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata >`:



## Public Types

- typedef `_Alloc::template rebind< _ATraits >` **\_\_rebind\_at**
- typedef `_Alloc::template rebind< \_Node\_base >` **\_\_rebind\_n**
- typedef `_ATraits::const_iterator` **a\_const\_iterator**
- typedef `__rebind_at::other::const_pointer` **a\_const\_pointer**
- typedef `_ATraits` **access\_traits**
- typedef `_Alloc` **allocator\_type**

- typedef \_\_rebind\_n::other::pointer **node\_pointer**
- typedef \_ATraits::type\_traits **type\_traits**

#### Public Member Functions

- **\_Node\_base** ([node\\_type](#) type)

#### Public Attributes

- node\_pointer **m\_p\_parent**
- const [node\\_type](#) **m\_type**

#### 4.275.1 Detailed Description

template<typename \_ATraits, typename Metadata>struct \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_base< \_ATraits, Metadata >

Node base.

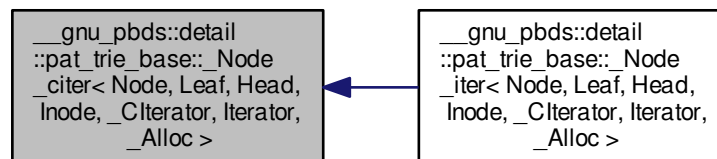
Definition at line 92 of file pat\_trie\_base.hpp.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

#### 4.276 \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer< Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc > Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer< Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc >:



#### Public Types

- typedef \_Alloc::template rebind< [metadata\\_type](#) > [\\_\\_rebind\\_m](#)
- typedef \_\_rebind\_m::other [\\_\\_rebind\\_ma](#)
- typedef value\_type **const\_reference**
- typedef [trivial\\_iterator\\_difference\\_type](#) **difference\_type**
- typedef [trivial\\_iterator\\_tag](#) **iterator\_category**

- typedef `__rebind_ma::const_reference` **metadata\_const\_reference**
- typedef `Node::metadata_type` **metadata\_type**
- typedef `value_type` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `_Citerator` **value\_type**

#### Public Member Functions

- **\_Node\_citer** (node\_pointer p\_nd=0, a\_const\_pointer p\_traits=0)
- **\_Node\_citer** `get_child` (size\_type i) const
- metadata\_const\_reference `get_metadata` () const
- size\_type `num_children` () const
- bool `operator!=` (const **\_Node\_citer** &other) const
- const\_reference `operator*` () const
- bool `operator==` (const **\_Node\_citer** &other) const
- `std::pair< a_const_iterator, a_const_iterator >` `valid_prefix` () const

#### Public Attributes

- node\_pointer **m\_p\_nd**
- a\_const\_pointer **m\_p\_traits**

#### Protected Types

- typedef `_Alloc::template rebind< Inode >` **\_\_rebind\_in**
- typedef `_Alloc::template rebind< Leaf >` **\_\_rebind\_l**
- typedef `_Alloc::template rebind< Node >` **\_\_rebind\_n**
- typedef `Node::a_const_iterator` **a\_const\_iterator**
- typedef `Node::a_const_pointer` **a\_const\_pointer**
- typedef `__rebind_in::other::const_pointer` **inode\_const\_pointer**
- typedef `__rebind_in::other::pointer` **inode\_pointer**
- typedef `__rebind_l::other::const_pointer` **leaf\_const\_pointer**
- typedef `__rebind_l::other::pointer` **leaf\_pointer**
- typedef `__rebind_n::other::pointer` **node\_pointer**

#### 4.276.1 Detailed Description

`template<typename Node, typename Leaf, typename Head, typename Inode, typename _Citerator, typename Iterator, typename _Alloc> class __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _Citerator, Iterator, _Alloc >`

Node const iterator.

Definition at line 814 of file `pat_trie_base.hpp`.

#### 4.276.2 Member Typedef Documentation

4.276.2.1 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > typedef _Alloc::template rebind<metadata_type> __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::_rebind_m`

Const metadata reference type.

Definition at line 869 of file pat\_trie\_base.hpp.

4.276.2.2 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > typedef Node::metadata_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::metadata_type`

Metadata type.

Definition at line 866 of file pat\_trie\_base.hpp.

#### 4.276.3 Member Function Documentation

4.276.3.1 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > _Node_citer __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::get_child ( size_type i ) const [inline]`

Returns a \_\_const node \_\_iterator to the corresponding node's i-th child.

Definition at line 911 of file pat\_trie\_base.hpp.

References std::advance().

4.276.3.2 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > metadata_const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::get_metadata ( ) const [inline]`

Metadata access.

Definition at line 894 of file pat\_trie\_base.hpp.

4.276.3.3 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > size_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::num_children ( ) const [inline]`

Returns the number of children in the corresponding node.

Definition at line 899 of file pat\_trie\_base.hpp.

References std::distance().

Referenced by \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer< Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc >::operator\*(), and \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_iter< Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc >::operator\*().

4.276.3.4 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::operator!= ( const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other ) const [inline]`

Compares content (negatively) to a different iterator object.

Definition at line 927 of file `pat_trie_base.hpp`.

4.276.3.5 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::operator*( ) const [inline]`

Const access; returns the `__const iterator*` associated with the current leaf.

Definition at line 886 of file `pat_trie_base.hpp`.

References `__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::num_children()`.

4.276.3.6 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::operator==( const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other ) const [inline]`

Compares content to a different iterator object.

Definition at line 922 of file `pat_trie_base.hpp`.

4.276.3.7 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > std::pair<a_const_iterator, a_const_iterator> __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::valid_prefix ( ) const [inline]`

Subtree valid prefix.

Definition at line 880 of file `pat_trie_base.hpp`.

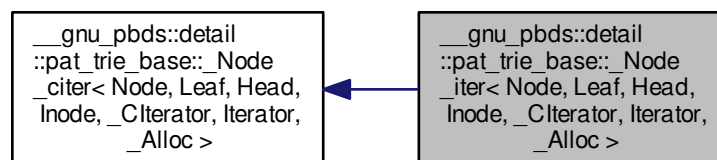
References `std::make_pair()`.

The documentation for this class was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 4.277 `__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >`:



## Public Types

- typedef \_\_Alloc::template rebind< [metadata\\_type](#) > [\\_\\_rebind\\_m](#)
- typedef \_\_rebind\_m::other [\\_\\_rebind\\_ma](#)
- typedef value\_type **const\_reference**
- typedef [trivial\\_iterator\\_difference\\_type](#) **difference\_type**
- typedef [trivial\\_iterator\\_tag](#) **iterator\_category**
- typedef \_\_rebind\_ma::const\_reference **metadata\_const\_reference**
- typedef Node::metadata\_type [metadata\\_type](#)
- typedef value\_type **reference**
- typedef base\_type::size\_type **size\_type**
- typedef Iterator **value\_type**

## Public Member Functions

- [\\_Node\\_iter](#) (node\_pointer p\_nd=0, a\_const\_pointer p\_traits=0)
- [\\_Node\\_iter](#) [get\\_child](#) (size\_type i) const
- metadata\_const\_reference [get\\_metadata](#) () const
- size\_type [num\\_children](#) () const
- bool [operator!=](#) (const [\\_Node\\_citer](#) &other) const
- reference [operator\\*](#) () const
- bool [operator==](#) (const [\\_Node\\_citer](#) &other) const
- [std::pair](#)< a\_const\_iterator, a\_const\_iterator > [valid\\_prefix](#) () const

## Public Attributes

- node\_pointer **m\_p\_nd**
- a\_const\_pointer **m\_p\_traits**

## Protected Types

- typedef \_\_Alloc::template rebind< Inode > [\\_\\_rebind\\_in](#)
- typedef \_\_Alloc::template rebind< Leaf > [\\_\\_rebind\\_l](#)
- typedef Node::a\_const\_iterator **a\_const\_iterator**
- typedef \_\_rebind\_in::other::const\_pointer **inode\_const\_pointer**
- typedef \_\_rebind\_l::other::const\_pointer **leaf\_const\_pointer**
- typedef \_\_rebind\_l::other::pointer **leaf\_pointer**

## 4.277.1 Detailed Description

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename Iterator, typename _C↵
Alloc>class __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >
```

Node iterator.

Definition at line 943 of file pat\_trie\_base.hpp.

#### 4.277.2 Member Typedef Documentation

4.277.2.1 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator  
, typename Iterator , typename _Alloc > typedef _Alloc::template rebind<metadata_type>  
__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc  
>::_rebind_m [inherited]`

Const metadata reference type.

Definition at line 869 of file `pat_trie_base.hpp`.

4.277.2.2 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator ,  
typename _Alloc > typedef Node::metadata_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf,  
Head, Inode, _CIterator, Iterator, _Alloc >::metadata_type [inherited]`

Metadata type.

Definition at line 866 of file `pat_trie_base.hpp`.

#### 4.277.3 Member Function Documentation

4.277.3.1 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator ,  
typename _Alloc > _Node_iter __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode,  
_CIterator, Iterator, _Alloc >::get_child ( size_type i ) const [inline]`

Returns a node `__iterator` to the corresponding node's `i`-th child.

Definition at line 976 of file `pat_trie_base.hpp`.

References `std::advance()`, and `std::begin()`.

4.277.3.2 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator ,  
typename _Alloc > metadata_const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf,  
Head, Inode, _CIterator, Iterator, _Alloc >::get_metadata ( ) const [inline],[inherited]`

Metadata access.

Definition at line 894 of file `pat_trie_base.hpp`.

4.277.3.3 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator  
, typename _Alloc > size_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode,  
_CIterator, Iterator, _Alloc >::num_children ( ) const [inline],[inherited]`

Returns the number of children in the corresponding node.

Definition at line 899 of file `pat_trie_base.hpp`.

References `std::distance()`.

Referenced by `__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc  
>::operator*()`, and `__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc  
>::operator*()`.

4.277.3.4 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator ,  
typename _Alloc > bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator,  
Iterator, _Alloc >::operator!=( const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other )  
const [inline],[inherited]`

Compares content (negatively) to a different iterator object.



Definition at line 927 of file pat\_trie\_base.hpp.

```
4.277.3.5  template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator ,
            typename _Alloc > reference __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode,
            _CIterator, Iterator, _Alloc >::operator* ( ) const    [inline]
```

Access; returns the iterator\* associated with the current leaf.

Definition at line 968 of file pat\_trie\_base.hpp.

References \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer< Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc >↵  
::num\_children().

```
4.277.3.6  template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator ,
            typename _Alloc > bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator,
            Iterator, _Alloc >::operator== ( const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other )
            const    [inline],[inherited]
```

Compares content to a different iterator object.

Definition at line 922 of file pat\_trie\_base.hpp.

```
4.277.3.7  template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename
            Iterator , typename _Alloc > std::pair<a_const_iterator, a_const_iterator> __gnu_pbds::detail::pat_trie_↵
            base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::valid_prefix ( ) const    [inline],
            [inherited]
```

Subtree valid prefix.

Definition at line 880 of file pat\_trie\_base.hpp.

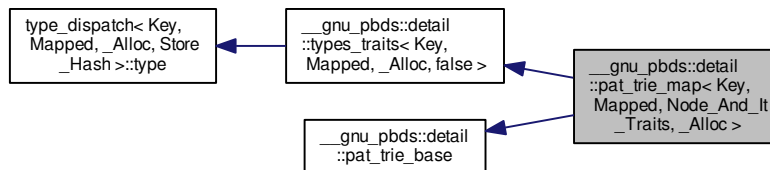
References std::make\_pair().

The documentation for this class was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 4.278 \_\_gnu\_pbds::detail::pat\_trie\_map< Key, Mapped, Node\_And\_It\_Traits, \_Alloc > Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::pat\_trie\_map< Key, Mapped, Node\_And\_It\_Traits, \_Alloc >:



### Public Types

- typedef traits\_type::access\_traits **access\_traits**

- typedef `_Alloc` **allocator\_type**
- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `point_const_iterator` **const\_iterator**
- typedef `traits_base::const_pointer` **const\_pointer**
- typedef `traits_base::const_reference` **const\_reference**
- typedef `traits_type::const_reverse_iterator` **const\_reverse\_iterator**
- typedef `pat_trie_tag` **container\_category**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `point_iterator` **iterator**
- typedef `traits_base::key_const_pointer` **key\_const\_pointer**
- typedef `traits_base::key_const_reference` **key\_const\_reference**
- typedef `traits_base::key_pointer` **key\_pointer**
- typedef `traits_base::key_reference` **key\_reference**
- typedef `traits_base::key_type` **key\_type**
- typedef `traits_base::mapped_const_pointer` **mapped\_const\_pointer**
- typedef `traits_base::mapped_const_reference` **mapped\_const\_reference**
- typedef `traits_base::mapped_pointer` **mapped\_pointer**
- typedef `traits_base::mapped_reference` **mapped\_reference**
- typedef `traits_base::mapped_type` **mapped\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**
- typedef `traits_type::node_const_iterator` **node\_const\_iterator**
- typedef `traits_type::node_iterator` **node\_iterator**
- enum `node_type` { **i\_node**, **leaf\_node**, **head\_node** }
- typedef `traits_type::node_update` **node\_update**
- typedef `traits_type::const_iterator` **point\_const\_iterator**
- typedef `traits_type::iterator` **point\_iterator**
- typedef `traits_base::pointer` **pointer**
- typedef `traits_base::reference` **reference**
- typedef `traits_type::reverse_iterator` **reverse\_iterator**
- typedef `_Alloc::size_type` **size\_type**
- typedef `integral_constant< int, Store_Hash >` **store\_extra**
- typedef `traits_base::value_type` **value\_type**

#### Public Member Functions

- **pat\_trie\_map** (const access\_traits &)
- **pat\_trie\_map** (const `pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >` &)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- bool **erase** (key\_const\_reference)
- const\_iterator **erase** (const\_iterator)
- iterator **erase** (iterator)
- const\_reverse\_iterator **erase** (const\_reverse\_iterator)
- reverse\_iterator **erase** (reverse\_iterator)
- template<typename Pred > size\_type **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference)

- point\_const\_iterator **find** (key\_const\_reference) const
- access\_traits & **get\_access\_traits** ()
- const access\_traits & **get\_access\_traits** () const
- node\_update & **get\_node\_update** ()
- const node\_update & **get\_node\_update** () const
- [std::pair](#)< point\_iterator, bool > **insert** (const\_reference)
- void **join** ([pat\\_trie\\_map](#)< Key, Mapped, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **lower\_bound** (key\_const\_reference)
- point\_const\_iterator **lower\_bound** (key\_const\_reference) const
- size\_type **max\_size** () const
- node\_const\_iterator [node\\_begin](#) () const
- node\_iterator [node\\_begin](#) ()
- node\_const\_iterator [node\\_end](#) () const
- node\_iterator [node\\_end](#) ()
- mapped\_reference **operator[]** (key\_const\_reference r\_key)
- reverse\_iterator **rbegin** ()
- const\_reverse\_iterator **rbegin** () const
- reverse\_iterator **rend** ()
- const\_reverse\_iterator **rend** () const
- size\_type **size** () const
- void **split** (key\_const\_reference, [pat\\_trie\\_map](#)< Key, Mapped, Node\_And\_It\_Traits, \_Alloc > &)
- void **swap** ([pat\\_trie\\_map](#)< Key, Mapped, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **upper\_bound** (key\_const\_reference)
- point\_const\_iterator **upper\_bound** (key\_const\_reference) const

#### Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

#### Protected Member Functions

- template<typename It > void **copy\_from\_range** (It, It)
- node\_pointer **recursive\_copy\_node** (node\_const\_pointer)
- void **value\_swap** ([pat\\_trie\\_map](#)< Key, Mapped, Node\_And\_It\_Traits, \_Alloc > &)

#### 4.278.1 Detailed Description

template<typename Key, typename Mapped, typename Node\_And\_It\_Traits, typename \_Alloc>class [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_map](#)< Key, Mapped, Node\_And\_It\_Traits, \_Alloc >

PATRICIA trie.

This implementation loosely borrows ideas from: 1) Fast Mergeable Integer Maps, Okasaki, Gill 1998 2) Ptsset: Sets of integers implemented as Patricia trees, Jean-Christophe Filliatr, 2000.

Definition at line 101 of file [pat\\_trie\\_.hpp](#).

## 4.278.2 Member Enumeration Documentation

4.278.2.1 `enum __gnu_pbds::detail::pat_trie_base::node_type` `[inherited]`

Three types of nodes.

`i_node` is used by `_Inode`, `leaf_node` by `_Leaf`, and `head_node` by `_Head`.

Definition at line 58 of file `pat_trie_base.hpp`.

## 4.278.3 Member Function Documentation

4.278.3.1 `template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc> pat_trie_map<Key, Mapped, Node_And_It_Traits, _Alloc>::node_const_iterator __gnu_pbds::detail::pat_trie_map<Key, Mapped, Node_And_It_Traits, _Alloc>::node_begin( ) const` `[inline]`

Returns a `const_node_iterator` corresponding to the node at the root of the tree.

Definition at line 101 of file `pat_trie_.hpp`.

4.278.3.2 `template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc> pat_trie_map<Key, Mapped, Node_And_It_Traits, _Alloc>::node_iterator __gnu_pbds::detail::pat_trie_map<Key, Mapped, Node_And_It_Traits, _Alloc>::node_begin( )` `[inline]`

Returns a `node_iterator` corresponding to the node at the root of the tree.

Definition at line 107 of file `pat_trie_.hpp`.

4.278.3.3 `template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc> pat_trie_map<Key, Mapped, Node_And_It_Traits, _Alloc>::node_const_iterator __gnu_pbds::detail::pat_trie_map<Key, Mapped, Node_And_It_Traits, _Alloc>::node_end( ) const` `[inline]`

Returns a `const_node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 113 of file `pat_trie_.hpp`.

4.278.3.4 `template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc> pat_trie_map<Key, Mapped, Node_And_It_Traits, _Alloc>::node_iterator __gnu_pbds::detail::pat_trie_map<Key, Mapped, Node_And_It_Traits, _Alloc>::node_end( )` `[inline]`

Returns a `node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 119 of file `pat_trie_.hpp`.

The documentation for this class was generated from the following file:

- [pat\\_trie\\_.hpp](#)

4.279 `__gnu_pbds::detail::probe_fn_base<_Alloc>` Class Template Reference

## 4.279.1 Detailed Description

`template<typename _Alloc> class __gnu_pbds::detail::probe_fn_base<_Alloc>`

Probe functor base.

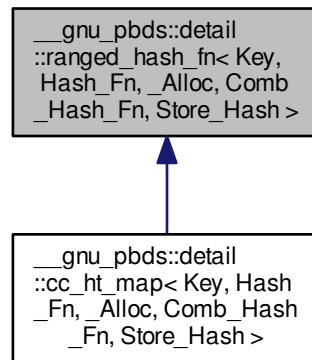
Definition at line 52 of file `probe_fn_base.hpp`.

The documentation for this class was generated from the following file:

- [probe\\_fn\\_base.hpp](#)

#### 4.280 `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >`:



##### 4.280.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Hash_Fn, bool Store_Hash>class __gnu_pbds<
::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >
```

Primary template.

Definition at line 55 of file `ranged_hash_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged\\_hash\\_fn.hpp](#)

#### 4.281 `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false >` Class Template Reference

Inherits `Hash_Fn`, and `Comb_Hash_Fn`.

##### Protected Types

- typedef `Comb_Hash_Fn` **comb\_hash\_fn\_base**
- typedef `Hash_Fn` **hash\_fn\_base**

- typedef `_Alloc::template rebind< Key >::other` **key\_allocator**
- typedef `key_allocator::const_reference` **key\_const\_reference**
- typedef `_Alloc::size_type` **size\_type**

#### Protected Member Functions

- **ranged\_hash\_fn** (size\_type)
- **ranged\_hash\_fn** (size\_type, const Hash\_Fn &)
- **ranged\_hash\_fn** (size\_type, const Hash\_Fn &, const Comb\_Hash\_Fn &)
- void **notify\_resized** (size\_type)
- size\_type **operator()** (key\_const\_reference) const
- void **swap** ([ranged\\_hash\\_fn](#)< Key, Hash\_Fn, \_Alloc, Comb\_Hash\_Fn, false > &)

#### 4.281.1 Detailed Description

template<typename Key, typename Hash\_Fn, typename \_Alloc, typename Comb\_Hash\_Fn>class `__gnu_pbds::detail::ranged_hash_fn`< Key, Hash\_Fn, \_Alloc, Comb\_Hash\_Fn, false >

Specialization 1 The client supplies a hash function and a ranged hash function, and requests that hash values not be stored.

Definition at line 71 of file `ranged_hash_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged\\_hash\\_fn.hpp](#)

#### 4.282 `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true >` Class Template Reference

Inherits `Hash_Fn`, and `Comb_Hash_Fn`.

#### Protected Types

- typedef `Comb_Hash_Fn` **comb\_hash\_fn\_base**
- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `Hash_Fn` **hash\_fn\_base**
- typedef `_Alloc::template rebind< Key >::other` **key\_allocator**
- typedef `key_allocator::const_reference` **key\_const\_reference**
- typedef `_Alloc::size_type` **size\_type**

#### Protected Member Functions

- **ranged\_hash\_fn** (size\_type)
- **ranged\_hash\_fn** (size\_type, const Hash\_Fn &)
- **ranged\_hash\_fn** (size\_type, const Hash\_Fn &, const Comb\_Hash\_Fn &)
- void **notify\_resized** (size\_type)
- [comp\\_hash](#) **operator()** (key\_const\_reference) const
- [comp\\_hash](#) **operator()** (key\_const\_reference, size\_type) const
- void **swap** ([ranged\\_hash\\_fn](#)< Key, Hash\_Fn, \_Alloc, Comb\_Hash\_Fn, true > &)

## 4.282.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Hash_Fn>class __gnu_pbds::detail::ranged_hash<
_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true >
```

Specialization 2 The client supplies a hash function and a ranged hash function, and requests that hash values be stored.

Definition at line 153 of file `ranged_hash_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged\\_hash\\_fn.hpp](#)

#### 4.283 `__gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false >` Class Template Reference

Inherits `Comb_Hash_Fn`.

##### Protected Types

- typedef `Comb_Hash_Fn` **comb\_hash\_fn\_base**
- typedef `_Alloc::size_type` **size\_type**

##### Protected Member Functions

- **ranged\_hash\_fn** (`size_type`)
- **ranged\_hash\_fn** (`size_type`, const `Comb_Hash_Fn` &)
- **ranged\_hash\_fn** (`size_type`, const [null\\_type](#) &, const `Comb_Hash_Fn` &)
- void **swap** ([ranged\\_hash\\_fn](#)< `Key`, [null\\_type](#), `_Alloc`, `Comb_Hash_Fn`, `false` > &)

## 4.283.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Comb_Hash_Fn>class __gnu_pbds::detail::ranged_hash_fn< Key, null_type,
_Alloc, Comb_Hash_Fn, false >
```

Specialization 3 The client does not supply a hash function (by specifying `null_type` as the `Hash_Fn` parameter), and requests that hash values not be stored.

Definition at line 255 of file `ranged_hash_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged\\_hash\\_fn.hpp](#)

#### 4.284 `__gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true >` Class Template Reference

Inherits `Comb_Hash_Fn`.

#### Protected Types

- typedef Comb\_Hash\_Fn **comb\_hash\_fn\_base**
- typedef \_Alloc::size\_type **size\_type**

#### Protected Member Functions

- **ranged\_hash\_fn** (size\_type)
- **ranged\_hash\_fn** (size\_type, const Comb\_Hash\_Fn &)
- **ranged\_hash\_fn** (size\_type, const [null\\_type](#) &, const Comb\_Hash\_Fn &)
- void **swap** ([ranged\\_hash\\_fn](#)< Key, [null\\_type](#), \_Alloc, Comb\_Hash\_Fn, true > &)

#### 4.284.1 Detailed Description

template<typename Key, typename \_Alloc, typename Comb\_Hash\_Fn>class `__gnu_pbds::detail::ranged_hash_fn`< Key, [null\\_type](#), \_Alloc, Comb\_Hash\_Fn, true >

Specialization 4 The client does not supply a hash function (by specifying `null_type` as the `Hash_Fn` parameter), and requests that hash values be stored.

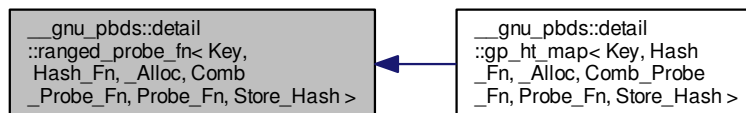
Definition at line 312 of file `ranged_hash_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged\\_hash\\_fn.hpp](#)

#### 4.285 `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash` > **Class Template Reference**

Inheritance diagram for `__gnu_pbds::detail::ranged_probe_fn`< Key, Hash\_Fn, \_Alloc, Comb\_Probe\_Fn, Probe\_Fn, Store\_Hash >:



#### 4.285.1 Detailed Description

template<typename Key, typename Hash\_Fn, typename \_Alloc, typename Comb\_Probe\_Fn, typename Probe\_Fn, bool Store\_Hash>class `__gnu_pbds::detail::ranged_probe_fn`< Key, Hash\_Fn, \_Alloc, Comb\_Probe\_Fn, Probe\_Fn, Store\_Hash >

Primary template.

Definition at line 55 of file `ranged_probe_fn.hpp`.

The documentation for this class was generated from the following file:



- [ranged\\_probe\\_fn.hpp](#)

#### 4.286 `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false >` Class Template Reference

Inherits `Hash_Fn`, `Comb_Probe_Fn`, and `Probe_Fn`.

##### Protected Types

- typedef `Comb_Probe_Fn` **comb\_probe\_fn\_base**
- typedef `Hash_Fn` **hash\_fn\_base**
- typedef `_Alloc::template rebind< Key >::other` **key\_allocator**
- typedef `key_allocator::const_reference` **key\_const\_reference**
- typedef `Probe_Fn` **probe\_fn\_base**
- typedef `_Alloc::size_type` **size\_type**

##### Protected Member Functions

- **ranged\_probe\_fn** (`size_type`)
- **ranged\_probe\_fn** (`size_type`, `const Hash_Fn &`)
- **ranged\_probe\_fn** (`size_type`, `const Hash_Fn &`, `const Comb_Probe_Fn &`)
- **ranged\_probe\_fn** (`size_type`, `const Hash_Fn &`, `const Comb_Probe_Fn &`, `const Probe_Fn &`)
- void **notify\_resized** (`size_type`)
- `size_type` **operator()** (`key_const_reference`) `const`
- `size_type` **operator()** (`key_const_reference`, `size_type`, `size_type`) `const`
- void **swap** ([ranged\\_probe\\_fn](#)< `Key`, `Hash_Fn`, `_Alloc`, `Comb_Probe_Fn`, `Probe_Fn`, `false` > &)

##### 4.286.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn>class __gnu_↵
pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false >
```

Specialization 1 The client supplies a probe function and a ranged probe function, and requests that hash values not be stored.

Definition at line 71 of file `ranged_probe_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged\\_probe\\_fn.hpp](#)

#### 4.287 `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true >` Class Template Reference

Inherits `Hash_Fn`, `Comb_Probe_Fn`, and `Probe_Fn`.

#### Protected Types

- typedef Comb\_Probe\_Fn **comb\_probe\_fn\_base**
- typedef [std::pair](#)< size\_type, size\_type > **comp\_hash**
- typedef Hash\_Fn **hash\_fn\_base**
- typedef \_Alloc::template rebind< Key >::other **key\_allocator**
- typedef key\_allocator::const\_reference **key\_const\_reference**
- typedef Probe\_Fn **probe\_fn\_base**
- typedef \_Alloc::size\_type **size\_type**

#### Protected Member Functions

- **ranged\_probe\_fn** (size\_type)
- **ranged\_probe\_fn** (size\_type, const Hash\_Fn &)
- **ranged\_probe\_fn** (size\_type, const Hash\_Fn &, const Comb\_Probe\_Fn &)
- **ranged\_probe\_fn** (size\_type, const Hash\_Fn &, const Comb\_Probe\_Fn &, const Probe\_Fn &)
- void **notify\_resized** (size\_type)
- **comp\_hash operator()** (key\_const\_reference) const
- size\_type **operator()** (key\_const\_reference, size\_type, size\_type) const
- size\_type **operator()** (key\_const\_reference, size\_type) const
- void **swap** ([ranged\\_probe\\_fn](#)< Key, Hash\_Fn, \_Alloc, Comb\_Probe\_Fn, Probe\_Fn, true > &)

#### 4.287.1 Detailed Description

template<typename Key, typename Hash\_Fn, typename \_Alloc, typename Comb\_Probe\_Fn, typename Probe\_Fn>class `__gnu_pbds::detail::ranged_probe_fn`< Key, Hash\_Fn, \_Alloc, Comb\_Probe\_Fn, Probe\_Fn, true >

Specialization 2- The client supplies a probe function and a ranged probe function, and requests that hash values not be stored.

Definition at line 176 of file `ranged_probe_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged\\_probe\\_fn.hpp](#)

#### 4.288 `__gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false >` Class Template Reference

Inherits Comb\_Probe\_Fn.

#### Protected Types

- typedef Comb\_Probe\_Fn **comb\_probe\_fn\_base**
- typedef \_Alloc::template rebind< Key >::other **key\_allocator**
- typedef key\_allocator::const\_reference **key\_const\_reference**
- typedef \_Alloc::size\_type **size\_type**

## Protected Member Functions

- **ranged\_probe\_fn** (size\_type size)
- **ranged\_probe\_fn** (size\_type, const Comb\_Probe\_Fn &r\_comb\_probe\_fn)
- **ranged\_probe\_fn** (size\_type, const [null\\_type](#) &, const Comb\_Probe\_Fn &r\_comb\_probe\_fn, const [null\\_type](#) &)
- void **swap** ([ranged\\_probe\\_fn](#) &other)

## 4.288.1 Detailed Description

template<typename Key, typename \_Alloc, typename Comb\_Probe\_Fn>class \_\_gnu\_pbds::detail::ranged\_probe\_fn< Key, null\_type, \_Alloc, Comb\_Probe\_Fn, null\_type, false >

Specialization 3 and 4 The client does not supply a hash function or probe function, and requests that hash values not be stored.

Definition at line 296 of file [ranged\\_probe\\_fn.hpp](#).

The documentation for this class was generated from the following file:

- [ranged\\_probe\\_fn.hpp](#)

#### 4.289 \_\_gnu\_pbds::detail::rb\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > Class Template Reference

Inherits [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_map< Key, Mapped, Cmp\\_Fn, Node\\_And\\_It\\_Traits, \\_Alloc >](#).

## Public Types

- typedef \_Alloc **allocator\_type**
- typedef Cmp\_Fn **cmp\_fn**
- typedef [std::pair](#)< size\_type, size\_type > **comp\_hash**
- typedef base\_type::const\_iterator **const\_iterator**
- typedef base\_type::const\_pointer **const\_pointer**
- typedef base\_type::const\_reference **const\_reference**
- typedef base\_type::const\_reverse\_iterator **const\_reverse\_iterator**
- typedef [rb\\_tree\\_tag](#) **container\_category**
- typedef \_Alloc::difference\_type **difference\_type**
- typedef base\_type::iterator **iterator**
- typedef base\_type::key\_const\_pointer **key\_const\_pointer**
- typedef base\_type::key\_const\_reference **key\_const\_reference**
- typedef base\_type::key\_pointer **key\_pointer**
- typedef base\_type::key\_reference **key\_reference**
- typedef base\_type::key\_type **key\_type**
- typedef base\_type::mapped\_const\_pointer **mapped\_const\_pointer**
- typedef base\_type::mapped\_const\_reference **mapped\_const\_reference**
- typedef base\_type::mapped\_pointer **mapped\_pointer**
- typedef base\_type::mapped\_reference **mapped\_reference**
- typedef base\_type::mapped\_type **mapped\_type**
- typedef \_\_nothrowcopy::indicator **no\_throw\_indicator**
- typedef traits\_type::const\_iterator **node\_const\_iterator**
- typedef traits\_type::node\_iterator **node\_iterator**

- `typedef base_type::node_update` **node\_update**
- `typedef base_type::const_iterator` **point\_const\_iterator**
- `typedef base_type::point_iterator` **point\_iterator**
- `typedef base_type::pointer` **pointer**
- `typedef base_type::reference` **reference**
- `typedef base_type::reverse_iterator` **reverse\_iterator**
- `typedef _Alloc::size_type` **size\_type**
- `typedef integral_constant< int, Store_Hash >` **store\_extra**
- `typedef base_type::value_type` **value\_type**

#### Public Member Functions

- **rb\_tree\_map** (const Cmp\_Fn &)
- **rb\_tree\_map** (const Cmp\_Fn &, const node\_update &)
- **rb\_tree\_map** (const [rb\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- `template<typename It > void` **copy\_from\_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- bool **erase** (key\_const\_reference)
- iterator **erase** (iterator)
- reverse\_iterator **erase** (reverse\_iterator)
- `template<typename Pred > size_type` **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference)
- point\_const\_iterator **find** (key\_const\_reference) const
- Cmp\_Fn & **get\_cmp\_fn** ()
- const Cmp\_Fn & **get\_cmp\_fn** () const
- `std::pair< point_iterator, bool >` **insert** (const\_reference)
- void **join** ([rb\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **lower\_bound** (key\_const\_reference)
- point\_const\_iterator **lower\_bound** (key\_const\_reference) const
- size\_type **max\_size** () const
- node\_const\_iterator **node\_begin** () const
- node\_iterator **node\_begin** ()
- node\_const\_iterator **node\_end** () const
- node\_iterator **node\_end** ()
- mapped\_reference **operator[]** (key\_const\_reference r\_key)
- reverse\_iterator **rbegin** ()
- const\_reverse\_iterator **rbegin** () const
- reverse\_iterator **rend** ()
- const\_reverse\_iterator **rend** () const
- size\_type **size** () const
- void **split** (key\_const\_reference, [rb\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **swap** ([rb\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **swap** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **upper\_bound** (key\_const\_reference)
- point\_const\_iterator **upper\_bound** (key\_const\_reference) const

### Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

### Protected Types

- typedef node\_allocator::value\_type **node**
- typedef \_Alloc::template rebind< typename traits\_type::node >::other **node\_allocator**
- typedef traits\_type::null\_node\_update\_pointer **null\_node\_update\_pointer**
- typedef [types\\_traits](#)< Key, Mapped, \_Alloc, false > **traits\_base**

### Protected Member Functions

- void **actual\_erase\_node** (node\_pointer)
- void **apply\_update** (node\_pointer, null\_node\_update\_pointer)
- template<typename Node\_Update\_ > void **apply\_update** (node\_pointer, Node\_Update\_ \*)
- [std::pair](#)< node\_pointer, bool > **erase** (node\_pointer)
- node\_pointer **get\_new\_node\_for\_leaf\_insert** (const\_reference, false\_type)
- node\_pointer **get\_new\_node\_for\_leaf\_insert** (const\_reference, true\_type)
- void **initialize\_min\_max** ()
- iterator **insert\_imp\_empty** (const\_reference)
- [std::pair](#)< point\_iterator, bool > **insert\_leaf** (const\_reference)
- iterator **insert\_leaf\_new** (const\_reference, node\_pointer, bool)
- void **join\_finish** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- bool **join\_prep** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- size\_type **recursive\_count** (node\_pointer) const
- void **rotate\_left** (node\_pointer)
- void **rotate\_parent** (node\_pointer)
- void **rotate\_right** (node\_pointer)
- void **split\_finish** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- bool **split\_prep** (key\_const\_reference, bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **update\_min\_max\_for\_erased\_node** (node\_pointer)
- void **update\_to\_top** (node\_pointer, null\_node\_update\_pointer)
- template<typename Node\_Update\_ > void **update\_to\_top** (node\_pointer, Node\_Update\_ \*)
- void **value\_swap** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)

### Static Protected Member Functions

- static void **clear\_imp** (node\_pointer)

### Protected Attributes

- node\_pointer **m\_p\_head**
- size\_type **m\_size**

### Static Protected Attributes

- static node\_allocator **s\_node\_allocator**

## 4.289.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>class __gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >
```

Red-Black tree.

This implementation uses an idea from the SGI STL (using a *header* node which is needed for efficient iteration).

Definition at line 84 of file `rb_tree_.hpp`.

## 4.289.2 Member Function Documentation

```
4.289.2.1 template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator
__gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin ( ) const
[inline], [inherited]
```

Returns a `const node_iterator` corresponding to the node at the root of the tree.

Definition at line 109 of file `bin_search_tree_.hpp`.

```
4.289.2.2 template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator
__gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin ( )
[inline], [inherited]
```

Returns a `node_iterator` corresponding to the node at the root of the tree.

Definition at line 117 of file `bin_search_tree_.hpp`.

```
4.289.2.3 template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator
__gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ( ) const
[inline], [inherited]
```

Returns a `const node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 125 of file `bin_search_tree_.hpp`.

```
4.289.2.4 template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator
__gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ( )
[inline], [inherited]
```

Returns a `node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 133 of file `bin_search_tree_.hpp`.

The documentation for this class was generated from the following file:

- [rb\\_tree\\_.hpp](#)

4.290 `__gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc >` Struct Template Reference

## Public Types

- typedef `_Alloc::template rebind< metadata_type >::other::const_reference` **metadata\_const\_reference**
- typedef `_Alloc::template rebind< metadata_type >::other::reference` **metadata\_reference**
- typedef `Metadata` **metadata\_type**
- typedef `_Alloc::template rebind< rb\_tree\_node\_< Value_Type, Metadata, _Alloc > >::other::pointer` **node\_←  
pointer**
- typedef `Value_Type` **value\_type**

## Public Member Functions

- `metadata_const_reference` **get\_metadata** () const
- `metadata_reference` **get\_metadata** ()
- `bool` **special** () const

## Public Attributes

- `metadata_type` **m\_metadata**
- `node_pointer` **m\_p\_left**
- `node_pointer` **m\_p\_parent**
- `node_pointer` **m\_p\_right**
- `bool` **m\_red**
- `value_type` **m\_value**

## 4.290.1 Detailed Description

`template<typename Value_Type, class Metadata, typename _Alloc>struct __gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc >`

Node for Red-Black trees.

Definition at line 52 of file `rb_tree_map_/node.hpp`.

The documentation for this struct was generated from the following file:

- [rb\\_tree\\_map\\_/node.hpp](#)

4.291 `__gnu_pbds::detail::rc< _Node, _Alloc >` Class Template Reference

## Public Types

- typedef `entry_const_pointer` **const\_iterator**
- typedef `node_pointer` **entry**

## Public Member Functions

- `rc` (const `rc` &)
- `const const_iterator` **begin** () const
- `void` **clear** ()
- `bool` **empty** () const

- `const const_iterator end () const`
- `void pop ()`
- `void push (entry)`
- `size_type size () const`
- `void swap (rc &)`
- `node_pointer top () const`

#### 4.291.1 Detailed Description

`template<typename _Node, typename _Alloc>class __gnu_pbds::detail::rc< _Node, _Alloc >`

Redundant binary counter.

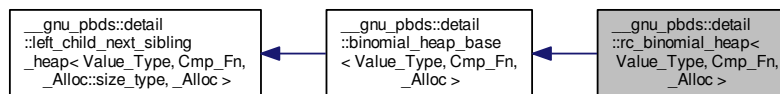
Definition at line 50 of file `rc.hpp`.

The documentation for this class was generated from the following file:

- [rc.hpp](#)

## 4.292 `__gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >`:



#### Public Types

- `typedef base_type::allocator_type allocator_type`
- `typedef base_type::cmp_fn cmp_fn`
- `typedef base_type::const_iterator const_iterator`
- `typedef base_type::const_pointer const_pointer`
- `typedef base_type::const_reference const_reference`
- `typedef _Alloc::difference_type difference_type`
- `typedef base_type::iterator iterator`
- `typedef base_type::point_const_iterator point_const_iterator`
- `typedef base_type::point_iterator point_iterator`
- `typedef base_type::pointer pointer`
- `typedef base_type::reference reference`
- `typedef _Alloc::size_type size_type`
- `typedef Value_Type value_type`



## Public Member Functions

- **rc\_binomial\_heap** (const Cmp\_Fn &)
- **rc\_binomial\_heap** (const [rc\\_binomial\\_heap](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- [iterator](#) **begin** ()
- [const\\_iterator](#) **begin** () const
- void **clear** ()
- bool **empty** () const
- [iterator](#) **end** ()
- [const\\_iterator](#) **end** () const
- void **erase** ([point\\_iterator](#))
- template<typename Pred > size\_type **erase\_if** (Pred)
- Cmp\_Fn & **get\_cmp\_fn** ()
- const Cmp\_Fn & **get\_cmp\_fn** () const
- void **join** ([rc\\_binomial\\_heap](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **join** ([binomial\\_heap\\_base](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- size\_type **max\_size** () const
- void **modify** ([point\\_iterator](#), const\_reference)
- void **pop** ()
- [point\\_iterator](#) **push** (const\_reference)
- size\_type **size** () const
- template<typename Pred > void **split** (Pred, [rc\\_binomial\\_heap](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- template<typename Pred > void **split** (Pred, [binomial\\_heap\\_base](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **swap** ([rc\\_binomial\\_heap](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **swap** ([left\\_child\\_next\\_sibling\\_heap](#)< Value\_Type, Cmp\_Fn, \_Alloc::size\_type, \_Alloc > &)
- const\_reference **top** () const

## Protected Types

- typedef base\_type::node **node**
- typedef \_Alloc::template rebind< [left\\_child\\_next\\_sibling\\_heap\\_node](#)< Value\_Type, \_Alloc::size\_type, \_Alloc > ::other **node\_allocator**
- typedef \_Alloc::size\_type **node\_metadata**
- typedef [std::pair](#)< node\_pointer, node\_pointer > **node\_pointer\_pair**

## Protected Member Functions

- void **actual\_erase\_node** (node\_pointer)
- void **bubble\_to\_top** (node\_pointer)
- void **clear\_imp** (node\_pointer)
- template<typename It > void **copy\_from\_range** (It, It)
- void **find\_max** ()
- node\_pointer **get\_new\_node\_for\_insert** (const\_reference)
- node\_pointer **prune** (Pred)
- void **swap** ([binomial\\_heap\\_base](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **swap\_with\_parent** (node\_pointer, node\_pointer)
- void **to\_linked\_list** ()
- void **value\_swap** ([left\\_child\\_next\\_sibling\\_heap](#) &)

## Static Protected Member Functions

- static void **make\_child\_of** (node\_pointer, node\_pointer)
- static node\_pointer **parent** (node\_pointer)

## Protected Attributes

- node\_pointer **m\_p\_max**
- node\_pointer **m\_p\_root**
- size\_type **m\_size**

## 4.292.1 Detailed Description

template<typename Value\_Type, typename Cmp\_Fn, typename \_Alloc>class `__gnu_pbds::detail::rc_binomial_heap`< Value\_Type, Cmp\_Fn, \_Alloc >

Redundant-counter binomial heap.

Definition at line 66 of file `rc_binomial_heap.hpp`.

The documentation for this class was generated from the following file:

- [rc\\_binomial\\_heap.hpp](#)

4.293 `__gnu_pbds::detail::resize_policy<_Tp>` Class Template Reference

## Public Types

- typedef `_Tp` **size\_type**

## Public Member Functions

- **resize\_policy** (const [resize\\_policy](#) &other)
- size\_type **get\_new\_size\_for\_arbitrary** (size\_type) const
- size\_type **get\_new\_size\_for\_grow** () const
- size\_type **get\_new\_size\_for\_shrink** () const
- bool **grow\_needed** (size\_type) const
- void **notify\_arbitrary** (size\_type)
- void **notify\_grow\_resize** ()
- void **notify\_shrink\_resize** ()
- bool **resize\_needed\_for\_grow** (size\_type) const
- bool **resize\_needed\_for\_shrink** (size\_type) const
- bool **shrink\_needed** (size\_type) const
- void **swap** ([resize\\_policy](#)<\_Tp> &)

## Static Public Attributes

- static const `_Tp` **min\_size**

## 4.293.1 Detailed Description

```
template<typename _Tp>class __gnu_pbds::detail::resize_policy< _Tp >
```

Resize policy for binary heap.

Definition at line 52 of file `resize_policy.hpp`.

The documentation for this class was generated from the following file:

- [resize\\_policy.hpp](#)

4.294 `__gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >` Class Template Reference

Inherits `__gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`.

## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `base_type::const_pointer` **const\_pointer**
- typedef `base_type::const_reference` **const\_reference**
- typedef `base_type::const_reverse_iterator` **const\_reverse\_iterator**
- typedef `splay_tree_tag` **container\_category**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::key_const_pointer` **key\_const\_pointer**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_pointer` **key\_pointer**
- typedef `base_type::key_reference` **key\_reference**
- typedef `base_type::key_type` **key\_type**
- typedef `base_type::mapped_const_pointer` **mapped\_const\_pointer**
- typedef `base_type::mapped_const_reference` **mapped\_const\_reference**
- typedef `base_type::mapped_pointer` **mapped\_pointer**
- typedef `base_type::mapped_reference` **mapped\_reference**
- typedef `base_type::mapped_type` **mapped\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**
- typedef `traits_type::node_const_iterator` **node\_const\_iterator**
- typedef `traits_type::node_iterator` **node\_iterator**
- typedef `base_type::node_update` **node\_update**
- typedef `base_type::const_iterator` **point\_const\_iterator**
- typedef `base_type::point_iterator` **point\_iterator**
- typedef `base_type::pointer` **pointer**
- typedef `base_type::reference` **reference**
- typedef `base_type::reverse_iterator` **reverse\_iterator**
- typedef `_Alloc::size_type` **size\_type**
- typedef `integral_constant< int, Store_Hash >` **store\_extra**
- typedef `base_type::value_type` **value\_type**

## Public Member Functions

- **splay\_tree\_map** (const Cmp\_Fn &)
- **splay\_tree\_map** (const Cmp\_Fn &, const node\_update &)
- **splay\_tree\_map** (const [splay\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- template<typename It > void **copy\_from\_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- bool **erase** (key\_const\_reference)
- iterator **erase** (iterator it)
- reverse\_iterator **erase** (reverse\_iterator)
- template<typename Pred > size\_type **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference)
- point\_const\_iterator **find** (key\_const\_reference) const
- Cmp\_Fn & **get\_cmp\_fn** ()
- const Cmp\_Fn & **get\_cmp\_fn** () const
- void **initialize** ()
- [std::pair](#)< point\_iterator, bool > **insert** (const\_reference r\_value)
- void **join** ([splay\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **lower\_bound** (key\_const\_reference)
- point\_const\_iterator **lower\_bound** (key\_const\_reference) const
- size\_type **max\_size** () const
- node\_const\_iterator **node\_begin** () const
- node\_iterator **node\_begin** ()
- node\_const\_iterator **node\_end** () const
- node\_iterator **node\_end** ()
- mapped\_reference **operator[]** (key\_const\_reference r\_key)
- reverse\_iterator **rbegin** ()
- const\_reverse\_iterator **rbegin** () const
- reverse\_iterator **rend** ()
- const\_reverse\_iterator **rend** () const
- size\_type **size** () const
- void **split** (key\_const\_reference, [splay\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **swap** ([splay\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **swap** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **upper\_bound** (key\_const\_reference)
- point\_const\_iterator **upper\_bound** (key\_const\_reference) const

## Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

### Protected Types

- typedef node\_allocator::value\_type **node**
- typedef \_Alloc::template rebind< typename traits\_type::node >::other **node\_allocator**
- typedef traits\_type::null\_node\_update\_pointer **null\_node\_update\_pointer**
- typedef [types\\_traits](#)< Key, Mapped, \_Alloc, false > **traits\_base**

### Protected Member Functions

- void **actual\_erase\_node** (node\_pointer)
- void **apply\_update** (node\_pointer, null\_node\_update\_pointer)
- template<typename Node\_Update\_ > void **apply\_update** (node\_pointer, Node\_Update\_ \*)
- [std::pair](#)< node\_pointer, bool > **erase** (node\_pointer)
- node\_pointer **get\_new\_node\_for\_leaf\_insert** (const\_reference, false\_type)
- node\_pointer **get\_new\_node\_for\_leaf\_insert** (const\_reference, true\_type)
- void **initialize\_min\_max** ()
- iterator **insert\_imp\_empty** (const\_reference)
- [std::pair](#)< point\_iterator, bool > **insert\_leaf** (const\_reference)
- iterator **insert\_leaf\_new** (const\_reference, node\_pointer, bool)
- void **join\_finish** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- bool **join\_prep** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- size\_type **recursive\_count** (node\_pointer) const
- void **rotate\_left** (node\_pointer)
- void **rotate\_parent** (node\_pointer)
- void **rotate\_right** (node\_pointer)
- void **split\_finish** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- bool **split\_prep** (key\_const\_reference, bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **update\_min\_max\_for\_erased\_node** (node\_pointer)
- void **update\_to\_top** (node\_pointer, null\_node\_update\_pointer)
- template<typename Node\_Update\_ > void **update\_to\_top** (node\_pointer, Node\_Update\_ \*)
- void **value\_swap** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)

### Static Protected Member Functions

- static void **clear\_imp** (node\_pointer)

### Protected Attributes

- node\_pointer **m\_p\_head**
- size\_type **m\_size**

### Static Protected Attributes

- static node\_allocator **s\_node\_allocator**

## 4.294.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>class __gnu_↵
pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >
```

Splay tree.

Definition at line 107 of file `splay_tree_.hpp`.

## 4.294.2 Member Function Documentation

4.294.2.1 `template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin ( ) const [inline], [inherited]`

Returns a `const node_iterator` corresponding to the node at the root of the tree.

Definition at line 109 of file `bin_search_tree_.hpp`.

4.294.2.2 `template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin ( ) [inline], [inherited]`

Returns a `node_iterator` corresponding to the node at the root of the tree.

Definition at line 117 of file `bin_search_tree_.hpp`.

4.294.2.3 `template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ( ) const [inline], [inherited]`

Returns a `const node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 125 of file `bin_search_tree_.hpp`.

4.294.2.4 `template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ( ) [inline], [inherited]`

Returns a `node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 133 of file `bin_search_tree_.hpp`.

The documentation for this class was generated from the following file:

- [splay\\_tree\\_.hpp](#)

4.295 `__gnu_pbds::detail::splay_tree_node_ < Value_Type, Metadata, _Alloc >` Struct Template Reference

## Public Types

- `typedef _Alloc::template rebind< metadata_type >::other::const_reference metadata_const_reference`

- typedef `_Alloc::template rebind< metadata_type >::other::reference` **metadata\_reference**
- typedef `Metadata` **metadata\_type**
- typedef `_Alloc::template rebind< splay\_tree\_node\_< Value_Type, Metadata, _Alloc > >::other::pointer` **node\_↵\_pointer**
- typedef `Value_Type` **value\_type**

#### Public Member Functions

- `metadata_const_reference` **get\_metadata** () const
- `metadata_reference` **get\_metadata** ()
- `bool` **special** () const

#### Public Attributes

- `metadata_type` **m\_metadata**
- `node_pointer` **m\_p\_left**
- `node_pointer` **m\_p\_parent**
- `node_pointer` **m\_p\_right**
- `bool` **m\_special**
- `value_type` **m\_value**

#### 4.295.1 Detailed Description

`template<typename Value_Type, class Metadata, typename _Alloc>struct __gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc >`

Node for splay tree.

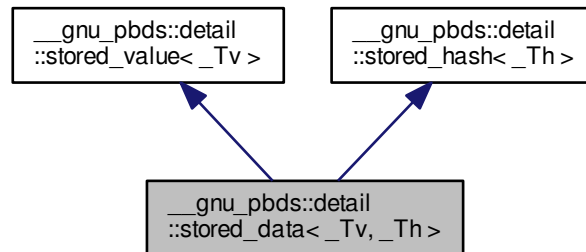
Definition at line 50 of file `splay_tree_/node.hpp`.

The documentation for this struct was generated from the following file:

- [splay\\_tree\\_/node.hpp](#)

4.296 `__gnu_pbds::detail::stored_data<_Tv,_Th>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::stored_data<_Tv,_Th>`:



#### Public Types

- typedef `_Th` **hash\_type**
- typedef `_Tv` **value\_type**

#### Public Attributes

- hash\_type **m\_hash**
- value\_type **m\_value**

#### 4.296.1 Detailed Description

```
template<typename _Tv, typename _Th>struct __gnu_pbds::detail::stored_data<_Tv,_Th>
```

Primary template for representation of stored data. Two types of data can be stored: value and hash.

Definition at line 95 of file `types_traits.hpp`.

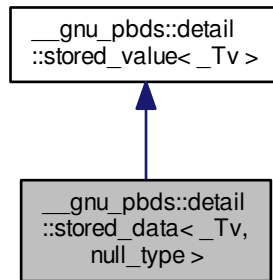
The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)



#### 4.297 `__gnu_pbds::detail::stored_data<_Tv, null_type>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::stored_data<_Tv, null_type>`:



##### Public Types

- `typedef _Tv value_type`

##### Public Attributes

- `value_type m_value`

##### 4.297.1 Detailed Description

```
template<typename _Tv>struct __gnu_pbds::detail::stored_data<_Tv, null_type>
```

Specialization for representation of stored data of just value type.

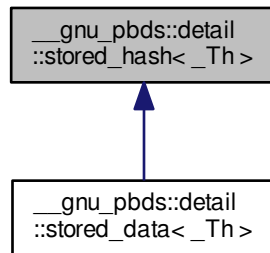
Definition at line 101 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

4.298 `__gnu_pbds::detail::stored_hash<_Th>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::stored_hash<_Th>`:



#### Public Types

- `typedef _Th hash_type`

#### Public Attributes

- `hash_type m_hash`

#### 4.298.1 Detailed Description

```
template<typename _Th> struct __gnu_pbds::detail::stored_hash<_Th>
```

Stored hash.

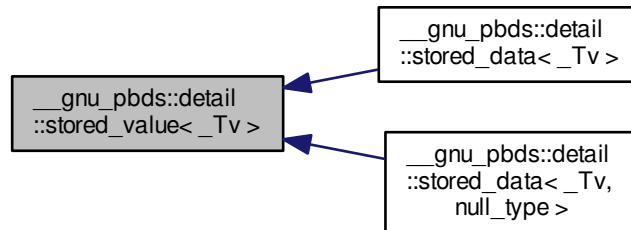
Definition at line 86 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

#### 4.299 `__gnu_pbds::detail::stored_value<_Tv>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::stored_value<_Tv>`:



##### Public Types

- typedef `_Tv` **value\_type**

##### Public Attributes

- value\_type **m\_value**

##### 4.299.1 Detailed Description

```
template<typename _Tv>struct __gnu_pbds::detail::stored_value<_Tv>
```

Stored value.

Definition at line 78 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

#### 4.300 `__gnu_pbds::detail::synth_access_traits<Type_Traits, Set, _ATraits>` Struct Template Reference

Inherits `_ATraits`.

##### Public Types

- typedef `_ATraits` **base\_type**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `type_traits::const_reference` **const\_reference**
- typedef `type_traits::key_const_reference` **key\_const\_reference**
- typedef `Type_Traits` **type\_traits**

## Public Member Functions

- **synth\_access\_traits** (const base\_type &)
- bool **cmp\_keys** (key\_const\_reference, key\_const\_reference) const
- bool **cmp\_prefixes** (const\_iterator, const\_iterator, const\_iterator, const\_iterator, bool compare\_after=false) const
- bool **equal\_keys** (key\_const\_reference, key\_const\_reference) const
- bool **equal\_prefixes** (const\_iterator, const\_iterator, const\_iterator, const\_iterator, bool compare\_after=true) const

## Static Public Member Functions

- static key\_const\_reference **extract\_key** (const\_reference)

## 4.300.1 Detailed Description

```
template<typename Type_Traits, bool Set, typename _ATraits>struct __gnu_pbds::detail::synth_access_traits< Type_Traits, Set, ↵
_ATraits >
```

Synthetic element access traits.

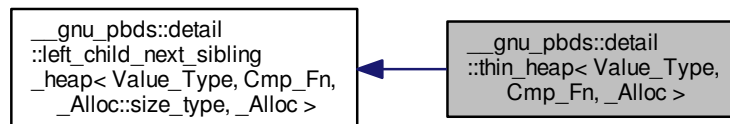
Definition at line 59 of file `synth_access_traits.hpp`.

The documentation for this struct was generated from the following file:

- [synth\\_access\\_traits.hpp](#)

4.301 `__gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >`:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `__rebind_a::const_pointer` **const\_pointer**
- typedef `__rebind_a::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**

- typedef [base\\_type::iterator](#) **iterator**
- typedef [base\\_type::point\\_const\\_iterator](#) **point\_const\_iterator**
- typedef [base\\_type::point\\_iterator](#) **point\_iterator**
- typedef [\\_\\_rebind\\_a::pointer](#) **pointer**
- typedef [\\_\\_rebind\\_a::reference](#) **reference**
- typedef [\\_Alloc::size\\_type](#) **size\_type**
- typedef [Value\\_Type](#) **value\_type**

#### Public Member Functions

- [iterator](#) **begin** ()
- [const\\_iterator](#) **begin** () const
- void **clear** ()
- bool **empty** () const
- [iterator](#) **end** ()
- [const\\_iterator](#) **end** () const
- void **erase** ([point\\_iterator](#))
- template<typename Pred > size\_type **erase\_if** (Pred)
- Cmp\_Fn & **get\_cmp\_fn** ()
- const Cmp\_Fn & **get\_cmp\_fn** () const
- void **join** ([thin\\_heap](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- size\_type **max\_size** () const
- void **modify** ([point\\_iterator](#), const\_reference)
- void **pop** ()
- [point\\_iterator](#) **push** (const\_reference)
- size\_type **size** () const
- template<typename Pred > void **split** (Pred, [thin\\_heap](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **swap** ([left\\_child\\_next\\_sibling\\_heap](#)< Value\_Type, Cmp\_Fn, \_Alloc::size\_type, \_Alloc > &)
- const\_reference **top** () const

#### Protected Types

- typedef [base\\_type::node](#) **node**
- typedef [\\_Alloc::template rebind< \[left\\\_child\\\_next\\\_sibling\\\_heap\\\_node\\\_\]\(#\)< Value\\_Type, \\_Alloc::size\\_type, \\_Alloc > ::other](#) **node\_allocator**
- typedef [base\\_type::node\\_const\\_pointer](#) **node\_const\_pointer**
- typedef [\\_Alloc::size\\_type](#) **node\_metadata**
- typedef [base\\_type::node\\_pointer](#) **node\_pointer**
- typedef [std::pair](#)< [node\\_pointer](#), [node\\_pointer](#) > **node\_pointer\_pair**

#### Protected Member Functions

- **thin\_heap** (const Cmp\_Fn &)
- **thin\_heap** (const [thin\\_heap](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **actual\_erase\_node** ([node\\_pointer](#))
- void **bubble\_to\_top** ([node\\_pointer](#))
- void **clear\_imp** ([node\\_pointer](#))
- template<typename It > void **copy\_from\_range** (It, It)
- [node\\_pointer](#) **get\_new\_node\_for\_insert** (const\_reference)
- [node\\_pointer](#) **prune** (Pred)

- void **swap** ([thin\\_heap](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **swap\_with\_parent** (node\_pointer, node\_pointer)
- void **to\_linked\_list** ()
- void **value\_swap** ([left\\_child\\_next\\_sibling\\_heap](#) &)

#### Static Protected Member Functions

- static node\_pointer **parent** (node\_pointer)

#### Protected Attributes

- node\_pointer **m\_p\_root**
- size\_type **m\_size**

#### 4.301.1 Detailed Description

`template<typename Value_Type, typename Cmp_Fn, typename _Alloc>class __gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >`

Thin heap.

See Tarjan and Kaplan.

Definition at line 77 of file `thin_heap.hpp`.

The documentation for this class was generated from the following file:

- [thin\\_heap.hpp](#)

### 4.302 `__gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp >` Struct Template Reference

#### 4.302.1 Detailed Description

`template<typename Node_Update, bool _BTp>struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp >`

Tree metadata helper.

Definition at line 58 of file `tree_policy/node_metadata_selector.hpp`.

The documentation for this struct was generated from the following file:

- [tree\\_policy/node\\_metadata\\_selector.hpp](#)

### 4.303 `__gnu_pbds::detail::tree_metadata_helper< Node_Update, false >` Struct Template Reference

#### Public Types

- typedef Node\_Update::metadata\_type **type**

#### 4.303.1 Detailed Description

```
template<typename Node_Update>struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, false >
```

Specialization, false.

Definition at line 62 of file tree\_policy/node\_metadata\_selector.hpp.

The documentation for this struct was generated from the following file:

- [tree\\_policy/node\\_metadata\\_selector.hpp](#)

#### 4.304 \_\_gnu\_pbds::detail::tree\_metadata\_helper< Node\_Update, true > Struct Template Reference

##### Public Types

- typedef [null\\_type](#) **type**

#### 4.304.1 Detailed Description

```
template<typename Node_Update>struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, true >
```

Specialization, true.

Definition at line 69 of file tree\_policy/node\_metadata\_selector.hpp.

The documentation for this struct was generated from the following file:

- [tree\\_policy/node\\_metadata\\_selector.hpp](#)

#### 4.305 \_\_gnu\_pbds::detail::tree\_node\_metadata\_dispatch< Key, Data, Cmp\_Fn, Node\_Update, \_Alloc > Struct Template Reference

##### Public Types

- typedef [tree\\_metadata\\_helper](#)< \_\_node\_u, null\_update >::type **type**

#### 4.305.1 Detailed Description

```
template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_Cltr, typename Const_Iterator, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>struct __gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >
```

Tree node metadata dispatch.

Definition at line 84 of file tree\_policy/node\_metadata\_selector.hpp.

The documentation for this struct was generated from the following file:

- [tree\\_policy/node\\_metadata\\_selector.hpp](#)

4.306 `__gnu_pbds::detail::tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc >` Struct Template Reference

## 4.306.1 Detailed Description

```
template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn,
typename _Alloc > class Node_Update, typename Tag, typename _Alloc> struct __gnu_pbds::detail::tree_traits< Key, Data,
Cmp_Fn, Node_Update, Tag, _Alloc >
```

Tree traits class, primary template.

Definition at line 70 of file `branch_policy/traits.hpp`.

The documentation for this struct was generated from the following file:

- [branch\\_policy/traits.hpp](#)

4.307 `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >` Struct Template Reference

## Public Types

- typedef `tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type` **metadata\_type**
- typedef `ov_tree_node_const_it_< value_type, metadata_type, _Alloc >` **node\_const\_iterator**
- typedef `ov_tree_node_it_< value_type, metadata_type, _Alloc >` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc >` **node\_update**
- typedef `__gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * null_node_update_pointer`

## 4.307.1 Detailed Description

```
template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class Cmp_Fn, typename
_Alloc_ > class Node_Update, typename _Alloc> struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_
tree_tag, _Alloc >
```

Tree traits.

Definition at line 61 of file `ov_tree_map_/traits.hpp`.

## 4.307.2 Member Typedef Documentation

4.307.2.1 `template<typename Key , typename Mapped , class Cmp_Fn , template< typename Node_Cltr, class Node_Itr, class Cmp_Fn, typename _Alloc_ > class Node_Update, typename _Alloc > typedef ov_tree_node_const_it_< value_type, metadata_type, _Alloc> __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >::node_const_iterator`

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 95 of file `ov_tree_map_/traits.hpp`.

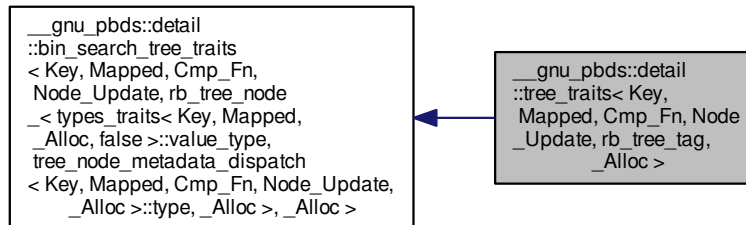
The documentation for this struct was generated from the following file:

- [ov\\_tree\\_map\\_/traits.hpp](#)



#### 4.308 `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`:



#### Public Types

- typedef `bin_search_tree_const_it_< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **const\_reverse\_iterator**
- typedef `rb_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >` **node**
- typedef `bin_search_tree_const_node_it_< rb_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc >` **node\_const\_iterator**
- typedef `bin_search_tree_node_it_< rb_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc >` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc >` **node\_update**
- typedef `__gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * null_node_update_pointer`
- typedef `bin_search_tree_const_it_< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point\_const\_iterator**
- typedef `bin_search_tree_it_< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point\_iterator**
- typedef `bin_search_tree_it_< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **reverse\_iterator**

#### 4.308.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn, typename _Alloc > class Node_Update, typename _Alloc> struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >
```

Specialization.

Definition at line 61 of file `rb_tree_map_/traits.hpp`.

#### 4.308.2 Member Typedef Documentation

4.308.2.1 `typedef bin_search_tree_const_node_it < rb_tree_node_ < types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc > , point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_node_ < types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc > , _Alloc >::node_const_iterator` `[inherited]`

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

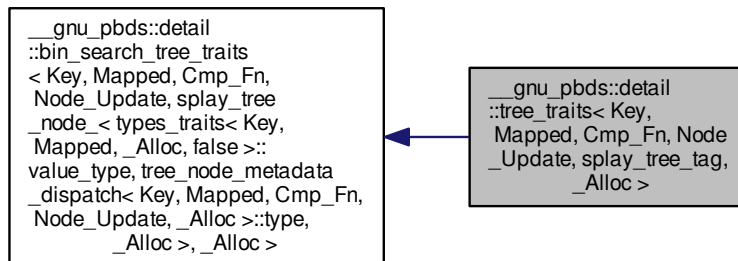
Definition at line 131 of file `bin_search_tree_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [rb\\_tree\\_map\\_/traits.hpp](#)

#### 4.309 `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`:



#### Public Types

- `typedef bin_search_tree_const_it < typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > const_reverse_iterator`
- `typedef splay_tree_node_ < types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc > node`
- `typedef bin_search_tree_const_node_it < splay_tree_node_ < types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc > node_const_iterator`

- typedef [bin\\_search\\_tree\\_node\\_it](#) < [splay\\_tree\\_node](#) < [types\\_traits](#) < Key, Mapped, \_Alloc, false >::value\_type, [tree\\_node\\_metadata\\_dispatch](#) < Key, Mapped, Cmp\_Fn, Node\_Update, \_Alloc >::type, \_Alloc >, [point\\_const\\_iterator](#), [point\\_iterator](#), \_Alloc > **node\_iterator**
- typedef Node\_Update < [node\\_const\\_iterator](#), [node\\_iterator](#), Cmp\_Fn, \_Alloc > **node\_update**
- typedef [\\_\\_gnu\\_pbds::null\\_node\\_update](#) < [node\\_const\\_iterator](#), [node\\_iterator](#), Cmp\_Fn, \_Alloc > \* **null\_node\_update\_pointer**
- typedef [bin\\_search\\_tree\\_const\\_it](#) < typename \_Alloc::template rebind< [node](#) >::other::pointer, typename type\_traits::value\_type, typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_traits::reference, typename type\_traits::const\_reference, true, \_Alloc > **point\_const\_iterator**
- typedef [bin\\_search\\_tree\\_it](#) < typename \_Alloc::template rebind< [node](#) >::other::pointer, typename type\_traits::value\_type, typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_traits::reference, typename type\_traits::const\_reference, true, \_Alloc > **point\_iterator**
- typedef [bin\\_search\\_tree\\_it](#) < typename \_Alloc::template rebind< [node](#) >::other::pointer, typename type\_traits::value\_type, typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_traits::reference, typename type\_traits::const\_reference, false, \_Alloc > **reverse\_iterator**

#### 4.309.1 Detailed Description

template<typename Key, typename Mapped, typename Cmp\_Fn, template< typename Node\_Cltr, typename Node\_Itr, typename Cmp\_Fn, typename \_Alloc > class Node\_Update, typename \_Alloc> struct [\\_\\_gnu\\_pbds::detail::tree\\_traits](#) < Key, Mapped, Cmp\_Fn, Node\_Update, [splay\\_tree\\_tag](#), \_Alloc >

Specialization.

Definition at line 61 of file [splay\\_tree\\_/traits.hpp](#).

#### 4.309.2 Member Typedef Documentation

4.309.2.1 typedef [bin\\_search\\_tree\\_const\\_node\\_it](#) < [splay\\_tree\\_node](#) < [types\\_traits](#) < Key, Mapped, \_Alloc, false >::value\_type, [tree\\_node\\_metadata\\_dispatch](#) < Key, Mapped, Cmp\_Fn, Node\_Update, \_Alloc >::type, \_Alloc >, [point\\_const\\_iterator](#), [point\\_iterator](#), \_Alloc> [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_traits](#) < Key, Mapped, Cmp\_Fn, Node\_Update, [splay\\_tree\\_node](#) < [types\\_traits](#) < Key, Mapped, \_Alloc, false >::value\_type, [tree\\_node\\_metadata\\_dispatch](#) < Key, Mapped, Cmp\_Fn, Node\_Update, \_Alloc >::type, \_Alloc >, \_Alloc >::**node\_const\_iterator** [inherited]

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 131 of file [bin\\_search\\_tree\\_/traits.hpp](#).

The documentation for this struct was generated from the following file:

- [splay\\_tree\\_/traits.hpp](#)

### 4.310 [\\_\\_gnu\\_pbds::detail::tree\\_traits](#) < Key, null\_type, Cmp\_Fn, Node\_Update, [ov\\_tree\\_tag](#), \_Alloc > Struct Template Reference

#### Public Types

- typedef [tree\\_node\\_metadata\\_dispatch](#) < Key, [null\\_type](#), Cmp\_Fn, Node\_Update, \_Alloc >::type **metadata\_type**
- typedef [ov\\_tree\\_node\\_const\\_it](#) < value\_type, metadata\_type, \_Alloc > [node\\_const\\_iterator](#)
- typedef [node\\_const\\_iterator](#) **node\_iterator**
- typedef Node\_Update < [node\\_const\\_iterator](#), [node\\_const\\_iterator](#), Cmp\_Fn, \_Alloc > **node\_update**
- typedef [\\_\\_gnu\\_pbds::null\\_node\\_update](#) < [node\\_const\\_iterator](#), [node\\_iterator](#), Cmp\_Fn, \_Alloc > \* **null\_node\_update\_pointer**

#### 4.310.1 Detailed Description

```
template<typename Key, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class Cmp_Fn_, typename _Alloc_ > class
Node_Update, typename _Alloc>struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc
>
```

Specialization.

Definition at line 132 of file `ov_tree_map_/traits.hpp`.

#### 4.310.2 Member Typedef Documentation

4.310.2.1 `template<typename Key , class Cmp_Fn , template< typename Node_Cltr, class Node_Itr, class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc > typedef ov_tree_node_const_it_< value_type, metadata_type, _Alloc> __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >::node_const_iterator`

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

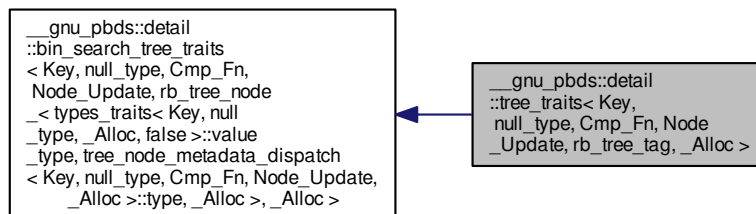
Definition at line 166 of file `ov_tree_map_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [ov\\_tree\\_map\\_/traits.hpp](#)

### 4.311 `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`:



#### Public Types

- typedef `bin_search_tree_const_it_< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **const\_reverse\_iterator**
- typedef `rb_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >` **node**

- `typedef bin_search_tree_const_node_it_< rb_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc > node_const_iterator`
- `typedef bin_search_tree_node_it_< rb_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc > node_iterator`
- `typedef Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > node_update`
- `typedef __gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * null_node_update_pointer`
- `typedef bin_search_tree_const_it_< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > point_const_iterator`
- `typedef bin_search_tree_it_< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > point_iterator`
- `typedef bin_search_tree_it_< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > reverse_iterator`

#### 4.311.1 Detailed Description

```
template<typename Key, typename Cmp_Fn, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn, typename _Alloc_ > class Node_Update, typename _Alloc> struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >
```

Specialization.

Definition at line 85 of file `rb_tree_map_/traits.hpp`.

#### 4.311.2 Member Typedef Documentation

- 4.311.2.1 `typedef bin_search_tree_const_node_it_< rb_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, _Alloc >::node_const_iterator` [inherited]

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

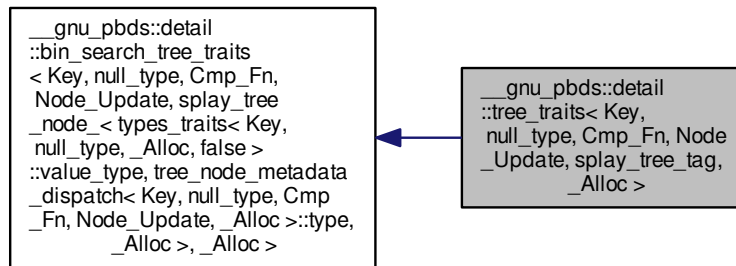
Definition at line 131 of file `bin_search_tree_/traits.hpp`.

The documentation for this struct was generated from the following file:

- `rb_tree_map_/traits.hpp`

## 4.312 `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`:



### Public Types

- typedef `bin_search_tree_const_it` < typename `_Alloc::template rebind< node >::other::pointer`, typename `type_traits::value_type`, typename `type_traits::pointer`, typename `type_traits::const_pointer`, typename `type_traits::reference`, typename `type_traits::const_reference`, false, `_Alloc` > **const\_reverse\_iterator**
- typedef `splay_tree_node` < `types_traits< Key, null_type, _Alloc, false >::value_type`, `tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type`, `_Alloc` > **node**
- typedef `bin_search_tree_const_node_it` < `splay_tree_node` < `types_traits< Key, null_type, _Alloc, false >::value_type`, `tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type`, `_Alloc` >, `point_const_iterator`, `point_iterator`, `_Alloc` > **node\_const\_iterator**
- typedef `bin_search_tree_node_it` < `splay_tree_node` < `types_traits< Key, null_type, _Alloc, false >::value_type`, `tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type`, `_Alloc` >, `point_const_iterator`, `point_iterator`, `_Alloc` > **node\_iterator**
- typedef `Node_Update` < `node_const_iterator`, `node_iterator`, `Cmp_Fn`, `_Alloc` > **node\_update**
- typedef `__gnu_pbds::null_node_update` < `node_const_iterator`, `node_iterator`, `Cmp_Fn`, `_Alloc` > \* **null\_node\_update\_pointer**
- typedef `bin_search_tree_const_it` < typename `_Alloc::template rebind< node >::other::pointer`, typename `type_traits::value_type`, typename `type_traits::pointer`, typename `type_traits::const_pointer`, typename `type_traits::reference`, typename `type_traits::const_reference`, true, `_Alloc` > **point\_const\_iterator**
- typedef `bin_search_tree_it` < typename `_Alloc::template rebind< node >::other::pointer`, typename `type_traits::value_type`, typename `type_traits::pointer`, typename `type_traits::const_pointer`, typename `type_traits::reference`, typename `type_traits::const_reference`, true, `_Alloc` > **point\_iterator**
- typedef `bin_search_tree_it` < typename `_Alloc::template rebind< node >::other::pointer`, typename `type_traits::value_type`, typename `type_traits::pointer`, typename `type_traits::const_pointer`, typename `type_traits::reference`, typename `type_traits::const_reference`, false, `_Alloc` > **reverse\_iterator**

### 4.312.1 Detailed Description

```
template<typename Key, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class Cmp_Fn, typename _Alloc_ > class
Node_Update, typename _Alloc>struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc
>
```

Specialization.

Definition at line 81 of file `splay_tree_/traits.hpp`.

#### 4.312.2 Member Typedef Documentation

4.312.2.1 `typedef bin_search_tree_const_node_it_< splay_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, _Alloc >::node_const_iterator` [inherited]

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 131 of file `bin_search_tree_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [splay\\_tree\\_/traits.hpp](#)

#### 4.313 \_\_gnu\_pbds::detail::trie\_metadata\_helper< Node\_Update, \_BTp > Struct Template Reference

##### 4.313.1 Detailed Description

```
template<typename Node_Update, bool _BTp>struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp >
```

Trie metadata helper.

Definition at line 58 of file `trie_policy/node_metadata_selector.hpp`.

The documentation for this struct was generated from the following file:

- [trie\\_policy/node\\_metadata\\_selector.hpp](#)

#### 4.314 \_\_gnu\_pbds::detail::trie\_metadata\_helper< Node\_Update, false > Struct Template Reference

##### Public Types

- `typedef Node_Update::metadata_type type`

##### 4.314.1 Detailed Description

```
template<typename Node_Update>struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, false >
```

Specialization, false.

Definition at line 62 of file `trie_policy/node_metadata_selector.hpp`.

The documentation for this struct was generated from the following file:

- [trie\\_policy/node\\_metadata\\_selector.hpp](#)

#### 4.315 `__gnu_pbds::detail::trie_metadata_helper< Node_Update, true >` Struct Template Reference

##### Public Types

- typedef `null_type` **type**

##### 4.315.1 Detailed Description

`template<typename Node_Update>struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, true >`

Specialization, true.

Definition at line 69 of file `trie_policy/node_metadata_selector.hpp`.

The documentation for this struct was generated from the following file:

- [trie\\_policy/node\\_metadata\\_selector.hpp](#)

#### 4.316 `__gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >` Struct Template Reference

##### Public Types

- typedef `trie_metadata_helper< __node_u, null_update >::type` **type**

##### 4.316.1 Detailed Description

`template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_Cltr, typename Const_Iterator, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>struct __gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >`

Trie node metadata dispatch.

Definition at line 84 of file `trie_policy/node_metadata_selector.hpp`.

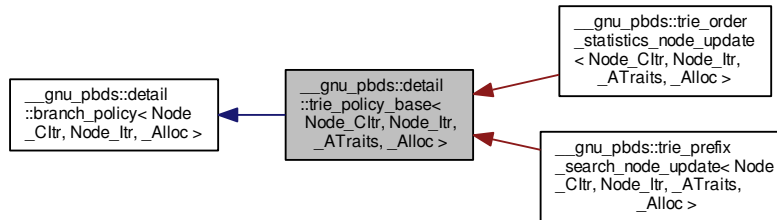
The documentation for this struct was generated from the following file:

- [trie\\_policy/node\\_metadata\\_selector.hpp](#)



#### 4.317 `__gnu_pbds::detail::trie_policy_base< Node_Cltr, Node_Itr, _ATraits, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::trie_policy_base< Node_Cltr, Node_Itr, _ATraits, _Alloc >`:



#### Public Types

- typedef `_ATraits` **access\_traits**
- typedef `_Alloc` **allocator\_type**
- typedef `node_const_iterator::value_type` **const\_iterator**
- typedef `node_iterator::value_type` **iterator**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_type` **key\_type**
- typedef `null_type` **metadata\_type**
- typedef `Node_Cltr` **node\_const\_iterator**
- typedef `Node_Itr` **node\_iterator**
- typedef `allocator_type::size_type` **size\_type**

#### Protected Types

- typedef `rebind_v::const_pointer` **const\_pointer**
- typedef `rebind_v::const_reference` **const\_reference**
- typedef `Node_Itr::value_type` **it\_type**
- typedef `remove_const< key_type >::type` **rckey\_type**
- typedef `remove_const< value_type >::type` **rcvalue\_type**
- typedef `_Alloc::template rebind< rkey_type >::other` **rebind\_k**
- typedef `_Alloc::template rebind< rcvalue_type >::other` **rebind\_v**
- typedef `rebind_v::reference` **reference**
- typedef `std::iterator_traits< it_type >::value_type` **value\_type**

#### Protected Member Functions

- virtual `const_iterator` **end** () const =0
- virtual `iterator` **end** ()=0
- `it_type` **end\_iterator** () const
- virtual `const access_traits &` **get\_access\_traits** () const =0
- virtual `node_const_iterator` **node\_begin** () const =0

- virtual node\_iterator **node\_begin** ()=0
- virtual node\_const\_iterator **node\_end** () const =0
- virtual node\_iterator **node\_end** ()=0

#### Static Protected Member Functions

- static size\_type **common\_prefix\_len** (node\_iterator, e\_const\_iterator, e\_const\_iterator, const access\_traits &)
- static key\_const\_reference **extract\_key** (const\_reference r\_val)
- static iterator **leftmost\_it** (node\_iterator)
- static bool **less** (e\_const\_iterator, e\_const\_iterator, e\_const\_iterator, e\_const\_iterator, const access\_traits &)
- static iterator **rightmost\_it** (node\_iterator)

#### 4.317.1 Detailed Description

template<typename Node\_Cltr, typename Node\_Itr, typename \_ATraits, typename \_Alloc>class `__gnu_pbds::detail::trie_policy_base`←  
base< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >

Base class for trie policies.

Definition at line 53 of file `trie_policy_base.hpp`.

The documentation for this class was generated from the following file:

- [trie\\_policy\\_base.hpp](#)

### 4.318 `__gnu_pbds::detail::trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc >` Struct Template Reference

#### 4.318.1 Detailed Description

template<typename Key, typename Data, typename \_ATraits, template< typename Node\_Cltr, typename Node\_Itr, typename \_A←  
Traits\_, typename \_Alloc > class Node\_Update, typename Tag, typename \_Alloc>struct `__gnu_pbds::detail::trie_traits`←  
\_ATraits, Node\_Update, Tag, \_Alloc >

Trie traits class, primary template.

Definition at line 83 of file `branch_policy/traits.hpp`.

The documentation for this struct was generated from the following file:

- [branch\\_policy/traits.hpp](#)

### 4.319 `__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >` Struct Template Reference

#### Public Types

- typedef \_ATraits **access\_traits**
- typedef [base\\_type::Cltr](#)< [node](#), [leaf](#), [head](#), [inode](#), true > **const\_iterator**
- typedef [base\\_type::Cltr](#)< [node](#), [leaf](#), [head](#), [inode](#), false > **const\_reverse\_iterator**
- typedef [base\\_type::Head](#)< [synth\\_access\\_traits](#), [metadata](#) > **head**
- typedef [base\\_type::Inode](#)< [synth\\_access\\_traits](#), [metadata](#) > **inode**

- typedef [base\\_type::lter](#)< [node](#), [leaf](#), [head](#), [inode](#), true > **iterator**
- typedef [base\\_type::Leaf](#)< [synth\\_access\\_traits](#), [metadata](#) > **leaf**
- typedef [base\\_type::Metadata](#)< [metadata\\_type](#), [\\_Alloc](#) > **metadata**
- typedef [trie\\_node\\_metadata\\_dispatch](#)< Key, Mapped, [\\_ATraits](#), [Node\\_Update](#), [\\_Alloc](#) >::type **metadata\_type**
- typedef [base\\_type::Node\\_base](#)< [synth\\_access\\_traits](#), [metadata](#) > **node**
- typedef [base\\_type::Node\\_citer](#)< [node](#), [leaf](#), [head](#), [inode](#), [const\\_iterator](#), [iterator](#), [\\_Alloc](#) > **node\_const\_iterator**
- typedef [base\\_type::Node\\_iter](#)< [node](#), [leaf](#), [head](#), [inode](#), [const\\_iterator](#), [iterator](#), [\\_Alloc](#) > **node\_iterator**
- typedef [Node\\_Update](#)< [node\\_const\\_iterator](#), [node\\_iterator](#), [\\_ATraits](#), [\\_Alloc](#) > **node\_update**
- typedef [null\\_node\\_update](#)< [node\\_const\\_iterator](#), [node\\_iterator](#), [\\_ATraits](#), [\\_Alloc](#) > \* **null\_node\_update\_pointer**
- typedef [base\\_type::lter](#)< [node](#), [leaf](#), [head](#), [inode](#), false > **reverse\_iterator**
- typedef [\\_\\_gnu\\_pbds::detail::synth\\_access\\_traits](#)< [type\\_traits](#), false, [access\\_traits](#) > [synth\\_access\\_traits](#)

#### 4.319.1 Detailed Description

```
template<typename Key, typename Mapped, typename _ATraits, template< typename Node_Cltr, typename Node_ltr, typename Cmp_Fn,
typename _Alloc_ > class Node_Update, typename _Alloc>struct __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits,
Node_Update, pat_trie_tag, _Alloc >
```

Specialization.

Definition at line 62 of file [pat\\_trie\\_/traits.hpp](#).

#### 4.319.2 Member Typedef Documentation

4.319.2.1 `template<typename Key , typename Mapped , typename _ATraits , template< typename Node_Cltr,
typename Node_ltr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc >
typedef base_type::Node_citer<node, leaf, head, inode, const_iterator, iterator, _Alloc>
__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc
>::node_const_iterator`

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 88 of file [pat\\_trie\\_/traits.hpp](#).

4.319.2.2 `template<typename Key , typename Mapped , typename _ATraits , template< typename Node_Cltr,
typename Node_ltr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc > typedef
Node_Update<node_const_iterator, node_iterator, _ATraits, _Alloc> __gnu_pbds::detail::trie_traits< Key,
Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_update`

Type for node update.

Definition at line 93 of file [pat\\_trie\\_/traits.hpp](#).

4.319.2.3 `template<typename Key , typename Mapped , typename _ATraits , template< typename Node_Cltr,
typename Node_ltr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc
> typedef __gnu_pbds::detail::synth_access_traits<type_traits, false, access_traits>
__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc
>::synth_access_traits`

Type for synthesized traits.

Definition at line 74 of file [pat\\_trie\\_/traits.hpp](#).

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_/traits.hpp](#)

## 4.320 `__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc > Struct` Template Reference

### Public Types

- typedef `_ATraits` **access\_traits**
- typedef `base_type::_Cltr< node, leaf, head, inode, true >` **const\_iterator**
- typedef `base_type::_Cltr< node, leaf, head, inode, false >` **const\_reverse\_iterator**
- typedef `base_type::_Head< synth_access_traits, metadata >` **head**
- typedef `base_type::_Inode< synth_access_traits, metadata >` **inode**
- typedef `const_iterator` **iterator**
- typedef `base_type::_Leaf< synth_access_traits, metadata >` **leaf**
- typedef `base_type::_Metadata< metadata_type, _Alloc >` **metadata**
- typedef `trie_node_metadata_dispatch< Key, null_type, _ATraits, Node_Update, _Alloc >::type` **metadata\_type**
- typedef `base_type::_Node_base< synth_access_traits, metadata >` **node**
- typedef `base_type::_Node_citer< node, leaf, head, inode, const_iterator, iterator, _Alloc >` **node\_const\_iterator**
- typedef `node_const_iterator` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, _ATraits, _Alloc >` **node\_update**
- typedef `null_node_update< node_const_iterator, node_const_iterator, _ATraits, _Alloc > *` **null\_node\_update**↵  
**\_pointer**
- typedef `const_reverse_iterator` **reverse\_iterator**
- typedef `__gnu_pbds::detail::synth_access_traits< type_traits, true, access_traits >` **synth\_access\_traits**

### 4.320.1 Detailed Description

```
template<typename Key, typename _ATraits, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc> struct __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >
```

Specialization.

Definition at line 109 of file `pat_trie_/traits.hpp`.

### 4.320.2 Member Typedef Documentation

4.320.2.1 `template<typename Key , typename _ATraits , template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc > typedef base_type::_Node_citer<node, leaf, head, inode, const_iterator, iterator, _Alloc> __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_const_iterator`

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 135 of file `pat_trie_/traits.hpp`.

4.320.2.2 `template<typename Key , typename _ATraits , template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc > typedef Node_Update<node_const_iterator, node_iterator, _ATraits, _Alloc> __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_update`

Type for node update.

Definition at line 140 of file `pat_trie_/traits.hpp`.

4.320.2.3 `template<typename Key , typename _ATraits , template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_,  
typename _Alloc_ > class Node_Update, typename _Alloc > typedef __gnu_pbds::detail::synth_access<  
_traits<type_traits, true, access_traits> __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits,  
Node_Update, pat_trie_tag, _Alloc >::synth_access_traits`

Type for synthesized traits.

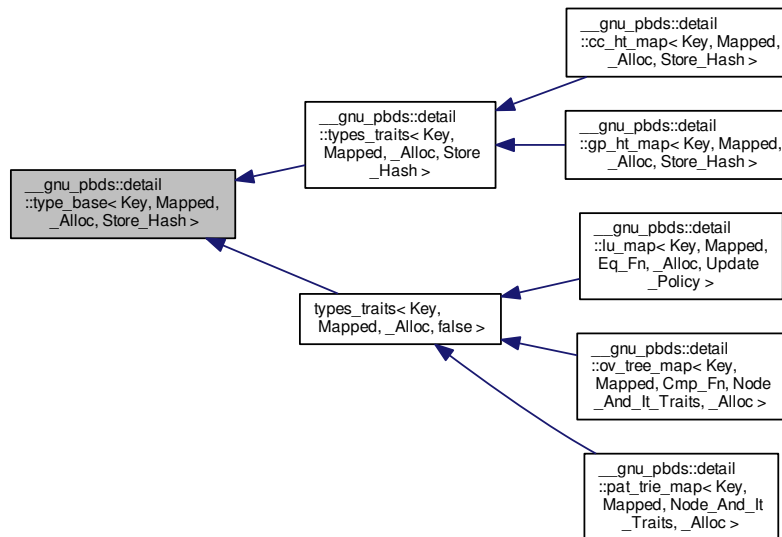
Definition at line 121 of file `pat_trie_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_/traits.hpp](#)

#### 4.321 \_\_gnu\_pbds::detail::type\_base< Key, Mapped, \_Alloc, Store\_Hash > Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, Store_Hash >`:



##### 4.321.1 Detailed Description

`template<typename Key, typename Mapped, typename _Alloc, bool Store_Hash>struct __gnu_pbds::detail::type_base< Key, Mapped, _Alloc, Store_Hash >`

Primary template.

Definition at line 107 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

4.322 `__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, false >` Struct Template Reference

## Public Types

- typedef `__rebind_va::const_pointer` **const\_pointer**
- typedef `__rebind_va::const_reference` **const\_reference**
- typedef `__rebind_ma::const_pointer` **mapped\_const\_pointer**
- typedef `__rebind_ma::const_reference` **mapped\_const\_reference**
- typedef `__rebind_ma::pointer` **mapped\_pointer**
- typedef `__rebind_ma::reference` **mapped\_reference**
- typedef `__rebind_ma::value_type` **mapped\_type**
- typedef `__rebind_va::pointer` **pointer**
- typedef `__rebind_va::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `stored_data< value_type, null_type >` **stored\_data\_type**
- typedef `__rebind_va::value_type` **value\_type**

## 4.322.1 Detailed Description

`template<typename Key, typename Mapped, typename _Alloc> struct __gnu_pbds::detail::type_base< Key, Mapped, _Alloc, false >`

Specialization of `type_base` for the case where the hash value is not stored alongside each value.

Definition at line 114 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

4.323 `__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, true >` Struct Template Reference

## Public Types

- typedef `__rebind_va::const_pointer` **const\_pointer**
- typedef `__rebind_va::const_reference` **const\_reference**
- typedef `__rebind_ma::const_pointer` **mapped\_const\_pointer**
- typedef `__rebind_ma::const_reference` **mapped\_const\_reference**
- typedef `__rebind_ma::pointer` **mapped\_pointer**
- typedef `__rebind_ma::reference` **mapped\_reference**
- typedef `__rebind_ma::value_type` **mapped\_type**
- typedef `__rebind_va::pointer` **pointer**
- typedef `__rebind_va::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `stored_data< value_type, size_type >` **stored\_data\_type**
- typedef `__rebind_va::value_type` **value\_type**

#### 4.323.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc>struct __gnu_pbds::detail::type_base< Key, Mapped, _Alloc, true >
```

Specialization of type\_base for the case where the hash value is stored alongside each value.

Definition at line 147 of file types\_traits.hpp.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

#### 4.324 \_\_gnu\_pbds::detail::type\_base< Key, null\_type, \_Alloc, false > Struct Template Reference

##### Public Types

- typedef \_\_rebind\_va::const\_pointer **const\_pointer**
- typedef \_\_rebind\_va::const\_reference **const\_reference**
- typedef \_\_rebind\_ma::const\_pointer **mapped\_const\_pointer**
- typedef \_\_rebind\_ma::const\_reference **mapped\_const\_reference**
- typedef \_\_rebind\_ma::pointer **mapped\_pointer**
- typedef \_\_rebind\_ma::reference **mapped\_reference**
- typedef \_\_rebind\_ma::value\_type **mapped\_type**
- typedef \_\_rebind\_va::pointer **pointer**
- typedef \_\_rebind\_va::reference **reference**
- typedef \_Alloc::size\_type **size\_type**
- typedef [stored\\_data](#)< value\_type, [null\\_type](#) > **stored\_data\_type**
- typedef Key **value\_type**

##### Static Public Attributes

- static [null\\_type](#) **s\_null\_type**

#### 4.324.1 Detailed Description

```
template<typename Key, typename _Alloc>struct __gnu_pbds::detail::type_base< Key, null_type, _Alloc, false >
```

Specialization of type\_base for the case where the hash value is not stored alongside each value.

Definition at line 181 of file types\_traits.hpp.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

#### 4.325 \_\_gnu\_pbds::detail::type\_base< Key, null\_type, \_Alloc, true > Struct Template Reference

##### Public Types

- typedef \_\_rebind\_va::const\_pointer **const\_pointer**
- typedef \_\_rebind\_va::const\_reference **const\_reference**
- typedef \_\_rebind\_ma::const\_pointer **mapped\_const\_pointer**

- typedef `__rebind_ma::const_reference` **mapped\_const\_reference**
- typedef `__rebind_ma::pointer` **mapped\_pointer**
- typedef `__rebind_ma::reference` **mapped\_reference**
- typedef `__rebind_ma::value_type` **mapped\_type**
- typedef `__rebind_va::pointer` **pointer**
- typedef `__rebind_va::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `stored_data< value_type, size_type >` **stored\_data\_type**
- typedef `Key` **value\_type**

#### Static Public Attributes

- static `null_type` **s\_null\_type**

##### 4.325.1 Detailed Description

`template<typename Key, typename _Alloc>struct __gnu_pbds::detail::type_base< Key, null_type, _Alloc, true >`

Specialization of `type_base` for the case where the hash value is stored alongside each value.

Definition at line 220 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

## 4.326 `__gnu_pbds::detail::type_dispatch< Key, Mapped, _Alloc, Store_Hash >` Struct Template Reference

#### Public Types

- typedef `type_base< Key, Mapped, _Alloc, Store_Hash >` **type**

##### 4.326.1 Detailed Description

`template<typename Key, typename Mapped, typename _Alloc, bool Store_Hash>struct __gnu_pbds::detail::type_dispatch< Key, Mapped, _Alloc, Store_Hash >`

Type base dispatch.

Definition at line 256 of file `types_traits.hpp`.

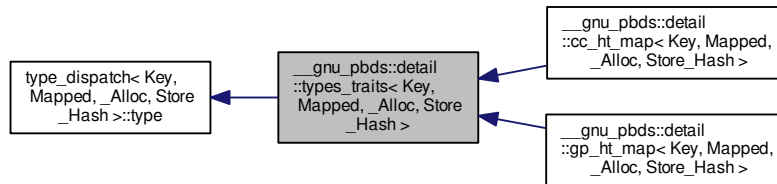
The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)



#### 4.327 `__gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >`:



##### Public Types

- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `__rebind_a::const_pointer` **key\_const\_pointer**
- typedef `__rebind_a::const_reference` **key\_const\_reference**
- typedef `__rebind_a::pointer` **key\_pointer**
- typedef `__rebind_a::reference` **key\_reference**
- typedef `__rebind_a::value_type` **key\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**
- typedef `_Alloc::size_type` **size\_type**
- typedef `integral_constant< int, Store_Hash >` **store\_extra**

##### Public Attributes

- `no_throw_indicator` **m\_no\_throw\_copies\_indicator**
- `store_extra` **m\_store\_extra\_indicator**

##### 4.327.1 Detailed Description

`template<typename Key, typename Mapped, typename _Alloc, bool Store_Hash>struct __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >`

Traits for abstract types.

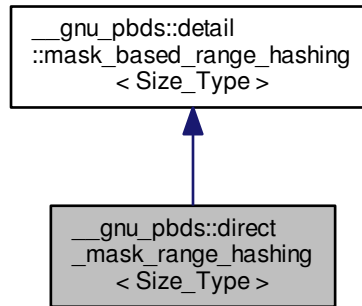
Definition at line 263 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

4.328 `__gnu_pbds::direct_mask_range_hashing< Size_Type >` Class Template Reference

Inheritance diagram for `__gnu_pbds::direct_mask_range_hashing< Size_Type >`:



## Public Types

- typedef `Size_Type` **size\_type**

## Public Member Functions

- void **swap** (`direct_mask_range_hashing< Size_Type >` &other)

## Protected Member Functions

- void **notify\_resized** (size\_type size)
- size\_type **operator()** (size\_type hash) const
- size\_type **range\_hash** (size\_type hash) const
- void **swap** (mask\_based\_range\_hashing &other)

## 4.328.1 Detailed Description

```
template<typename Size_Type = std::size_t>class __gnu_pbds::direct_mask_range_hashing< Size_Type >
```

A mask range-hashing class (uses a bitmask).

Definition at line 109 of file `hash_policy.hpp`.

## 4.328.2 Member Function Documentation

4.328.2.1 `template<typename Size_Type > direct_mask_range_hashing< Size_Type >::size_type __gnu_pbds::direct_mask_range_hashing< Size_Type >::operator() ( size_type hash ) const` `[inline]`, `[protected]`

Transforms the `__hash` value hash into a ranged-hash value (using a bit-mask).

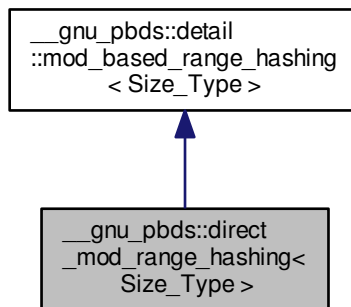
Definition at line 57 of file hash\_policy.hpp.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

#### 4.329 `__gnu_pbds::direct_mod_range_hashing< Size_Type >` Class Template Reference

Inheritance diagram for `__gnu_pbds::direct_mod_range_hashing< Size_Type >`:



##### Public Types

- typedef `Size_Type` **size\_type**

##### Public Member Functions

- void **swap** ([direct\\_mod\\_range\\_hashing](#)< `Size_Type` > &other)

##### Protected Member Functions

- void **notify\_resized** (size\_type size)
- size\_type [operator\(\)](#) (size\_type hash) const
- size\_type **range\_hash** (size\_type s) const
- void **swap** (mod\_based\_range\_hashing &other)

##### 4.329.1 Detailed Description

```
template<typename Size_Type = std::size_t>class __gnu_pbds::direct_mod_range_hashing< Size_Type >
```

A mod range-hashing class (uses the modulo function).

Definition at line 141 of file hash\_policy.hpp.

#### 4.329.2 Member Function Documentation

4.329.2.1 `template<typename Size_Type > direct_mod_range_hashing< Size_Type >::size_type  
__gnu_pbds::direct_mod_range_hashing< Size_Type >::operator() ( size_type hash ) const` `[inline]`,  
`[protected]`

Transforms the `__hash` value `hash` into a ranged-hash value (using a modulo operation).

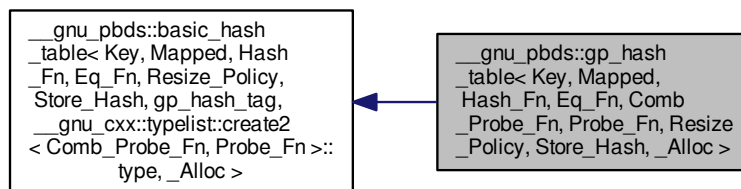
Definition at line 57 of file `hash_policy.hpp`.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

#### 4.330 `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >`:



#### Public Types

- typedef `Comb_Probe_Fn` **comb\_probe\_fn**
- typedef `gp_hash_tag` **container\_category**
- typedef `Eq_Fn` **eq\_fn**
- typedef `Hash_Fn` **hash\_fn**
- typedef `Probe_Fn` **probe\_fn**
- typedef `Resize_Policy` **resize\_policy**

#### Public Member Functions

- `gp_hash_table` ()
- `gp_hash_table` (const `hash_fn` &h)
- `gp_hash_table` (const `hash_fn` &h, const `eq_fn` &e)
- `gp_hash_table` (const `hash_fn` &h, const `eq_fn` &e, const `comb_probe_fn` &cp)
- `gp_hash_table` (const `hash_fn` &h, const `eq_fn` &e, const `comb_probe_fn` &cp, const `probe_fn` &p)
- `gp_hash_table` (const `hash_fn` &h, const `eq_fn` &e, const `comb_probe_fn` &cp, const `probe_fn` &p, const `resize_policy` &rp)

- `template<typename It > gp_hash_table` (It first, It last)
- `template<typename It > gp_hash_table` (It first, It last, const hash\_fn &h)
- `template<typename It > gp_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e)
- `template<typename It > gp_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp)
- `template<typename It > gp_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp, const probe\_fn &p)
- `template<typename It > gp_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp, const probe\_fn &p, const resize\_policy &rp)
- `gp_hash_table` (const `gp_hash_table` &other)
- `gp_hash_table` & `operator=` (const `gp_hash_table` &other)
- void `swap` (`gp_hash_table` &other)

#### 4.330.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn
= typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn
= typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<
Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> class __gnu_
pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >
```

A general-probing hash-based associative container.

##### Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Hash_Fn</i>	Hashing functor.
<i>Eq_Fn</i>	Equal functor.
<i>Comb_Probe_Fn</i>	Combining probe functor. If Hash_Fn is not null_type, then this is the ranged-probe functor; otherwise, this is the range-hashing functor. XXX See Design::Hash-Based Containers::Hash Policies.
<i>Probe_Fn</i>	Probe functor.
<i>Resize_Policy</i>	Resizes hash.
<i>Store_Hash</i>	Indicates whether the hash value will be stored along with each key. If Hash_Fn is null_type, then the container will not compile if this value is true
<i>_Alloc</i>	Allocator type.

Base tag choices are: `gp_hash_tag`.

Base is `basic_hash_table`.

Definition at line 368 of file `assoc_container.hpp`.

#### 4.330.2 Constructor & Destructor Documentation

4.330.2.1 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table ( ) [inline]`

Default constructor.

Definition at line 382 of file `assoc_container.hpp`.

```
4.330.2.2  template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
            typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
            detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type,
            typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash =
            detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped,
            Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table ( const hash_fn & h
            ) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object.

Definition at line 386 of file `assoc_container.hpp`.

```
4.330.2.3  template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
            typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
            detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type,
            typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash =
            detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped,
            Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table ( const hash_fn &
            h, const eq_fn & e ) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, and `r_eq_fn` will be copied by the `eq_fn` object of the container object.

Definition at line 393 of file `assoc_container.hpp`.

```
4.330.2.4  template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
            typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
            detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type,
            typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash =
            detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped,
            Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table ( const hash_fn &
            h, const eq_fn & e, const comb_probe_fn & cp ) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, and `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object.

Definition at line 401 of file `assoc_container.hpp`.

```
4.330.2.5  template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
            typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
            detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type,
            typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash =
            detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped,
            Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table ( const hash_fn &
            h, const eq_fn & e, const comb_probe_fn & cp, const probe_fn & p ) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object, and `r_probe_fn` will be copied by the `probe_fn` object of the container object.

Definition at line 410 of file `assoc_container.hpp`.

4.330.2.6 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table ( const hash_fn & h, const eq_fn & e, const comb_probe_fn & cp, const probe_fn & p, const resize_policy & rp ) [inline]`

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object, `r_probe_fn` will be copied by the `probe_fn` object of the container object, and `r_resize_policy` will be copied by the `Resize_Policy` object of the container object.

Definition at line 422 of file `assoc_container.hpp`.

4.330.2.7 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table ( It first, It last ) [inline]`

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

Definition at line 430 of file `assoc_container.hpp`.

4.330.2.8 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table ( It first, It last, const hash_fn & h ) [inline]`

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

Definition at line 438 of file `assoc_container.hpp`.

4.330.2.9 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table ( It first, It last, const hash_fn & h, const eq_fn & e ) [inline]`

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, and `r_eq_fn` will be copied by the `eq_fn` object of the container object.

Definition at line 449 of file `assoc_container.hpp`.

```
4.330.2.10 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type,
typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash
= detail::default_store_hash, typename _Alloc = std::allocator<char>>> template<typename It >
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table ( It first, It last, const hash_fn & h, const eq_fn & e, const comb_probe_fn &
cp ) [inline]
```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, and `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object.

Definition at line 461 of file `assoc_container.hpp`.

```
4.330.2.11 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type,
typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash
= detail::default_store_hash, typename _Alloc = std::allocator<char>>> template<typename It >
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table ( It first, It last, const hash_fn & h, const eq_fn & e, const comb_probe_fn &
cp, const probe_fn & p ) [inline]
```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object, and `r_probe_fn` will be copied by the `probe_fn` object of the container object.

Definition at line 475 of file `assoc_container.hpp`.

```
4.330.2.12 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type,
typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash
= detail::default_store_hash, typename _Alloc = std::allocator<char>>> template<typename It >
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table ( It first, It last, const hash_fn & h, const eq_fn & e, const comb_probe_fn &
cp, const probe_fn & p, const resize_policy & rp ) [inline]
```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_probe_fn` will be copied by the `comb_←probe_fn` object of the container object, `r_probe_fn` will be copied by the `probe_fn` object of the container object, and `r_resize_policy` will be copied by the `resize_policy` object of the container object.

Definition at line 491 of file `assoc_container.hpp`.

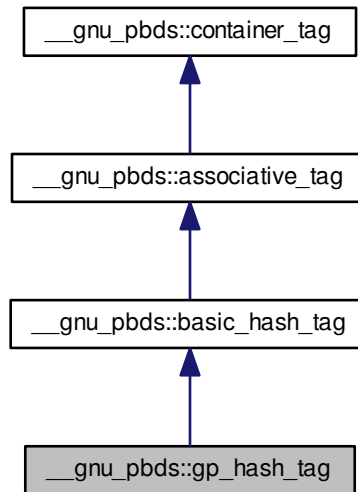
The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)



### 4.331 `__gnu_pbds::gp_hash_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::gp_hash_tag`:



#### 4.331.1 Detailed Description

General-probing hash.

Definition at line 144 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 4.332 `__gnu_pbds::hash_exponential_size_policy< Size_Type >` Class Template Reference

#### Public Types

- typedef `Size_Type` **size\_type**

#### Public Member Functions

- [hash\\_exponential\\_size\\_policy](#) (`size_type` start\_size=8, `size_type` grow\_factor=2)
- void **swap** ([hash\\_exponential\\_size\\_policy](#)< `Size_Type` > &other)

#### Protected Member Functions

- `size_type` **get\_nearest\_larger\_size** (`size_type` size) const
- `size_type` **get\_nearest\_smaller\_size** (`size_type` size) const

#### 4.332.1 Detailed Description

template<typename Size\_Type = std::size\_t>class `__gnu_pbds::hash_exponential_size_policy< Size_Type >`

A size policy whose sequence of sizes form an exponential sequence (typically powers of 2.

Definition at line 413 of file `hash_policy.hpp`.

#### 4.332.2 Constructor & Destructor Documentation

4.332.2.1 `template<typename Size_Type > __gnu_pbds::hash_exponential_size_policy< Size_Type >::hash_exponential_size_policy ( size_type start_size = 8, size_type grow_factor = 2 )`

Default constructor, or onstructor taking a `start_size`, or constructor taking a start size and `grow_factor`. The policy will use the sequence of sizes `start_size`, `start_size* grow_factor`, `start_size* grow_factor^2`, ...

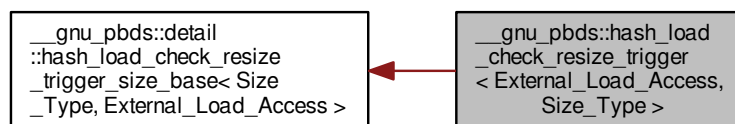
Definition at line 44 of file `hash_policy.hpp`.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

### 4.333 `__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >` Class Template Reference

Inheritance diagram for `__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >`:



#### Public Types

- enum { [external\\_load\\_access](#) }
- typedef Size\_Type **size\_type**

#### Public Member Functions

- [hash\\_load\\_check\\_resize\\_trigger](#) (float load\_min=0.125, float load\_max=0.5)
- `std::pair< float, float >` [get\\_loads](#) () const
- void [set\\_loads](#) (`std::pair< float, float >` load\_pair)
- void **swap** ([hash\\_load\\_check\\_resize\\_trigger](#) &other)

## Protected Member Functions

- bool **is\_grow\_needed** (size\_type size, size\_type num\_entries) const
- bool **is\_resize\_needed** () const
- void **notify\_cleared** ()
- void **notify\_erase\_search\_collision** ()
- void **notify\_erase\_search\_end** ()
- void **notify\_erase\_search\_start** ()
- void **notify\_erased** (size\_type num\_entries)
- void **notify\_externally\_resized** (size\_type new\_size)
- void **notify\_find\_search\_collision** ()
- void **notify\_find\_search\_end** ()
- void **notify\_find\_search\_start** ()
- void **notify\_insert\_search\_collision** ()
- void **notify\_insert\_search\_end** ()
- void **notify\_insert\_search\_start** ()
- void **notify\_inserted** (size\_type num\_entries)
- void **notify\_resized** (size\_type new\_size)

## 4.333.1 Detailed Description

```
template<bool External_Load_Access = false, typename Size_Type = std::size_t>class __gnu_pbds::hash_load_check_resize_↵
trigger< External_Load_Access, Size_Type >
```

A resize trigger policy based on a load check. It keeps the load factor between some load factors load\_min and load\_↵\_max.

Definition at line 175 of file hash\_policy.hpp.

## 4.333.2 Member Enumeration Documentation

4.333.2.1 `template<bool External_Load_Access = false, typename Size_Type = std::size_t> anonymous enum`

## Enumerator

**external\_load\_access** Specifies whether the load factor can be accessed externally. The two options have different trade-offs in terms of flexibility, genericity, and encapsulation.

Definition at line 180 of file hash\_policy.hpp.

## 4.333.3 Constructor &amp; Destructor Documentation

4.333.3.1 `template<bool External_Load_Access, typename Size_Type > __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::hash_load_check_resize_trigger ( float load_min = 0.125, float load_max = 0.5 )`

Default constructor, or constructor taking load\_min and load\_max load factors between which this policy will keep the actual load.

Definition at line 47 of file hash\_policy.hpp.

## 4.333.4 Member Function Documentation

4.333.4.1 `template<bool External_Load_Access, typename Size_Type > std::pair< float, float > __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::get_loads ( ) const`  
`[inline]`

Returns a pair of the minimal and maximal loads, respectively.

Definition at line 236 of file `hash_policy.hpp`.

4.333.4.2 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_cleared ( )`  
`[protected]`

Notifies the table was cleared.

Definition at line 206 of file `hash_policy.hpp`.

4.333.4.3 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_inserted ( size_type num_entries )` `[inline]`,  
`[protected]`

Notifies an element was inserted. The total number of entries in the table is `num_entries`.

Definition at line 109 of file `hash_policy.hpp`.

4.333.4.4 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_resized ( size_type new_size )`  
`[protected]`

Notifies the table was resized as a result of this object's signifying that a resize is needed.

Definition at line 151 of file `hash_policy.hpp`.

4.333.4.5 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::set_loads ( std::pair< float, float > load_pair )`

Sets the loads through a pair of the minimal and maximal loads, respectively.

Definition at line 245 of file `hash_policy.hpp`.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

4.334 `__gnu_pbds::hash_prime_size_policy` Class Reference

## Public Types

- typedef `std::size_t` [size\\_type](#)

## Public Member Functions

- [hash\\_prime\\_size\\_policy](#) ([size\\_type](#) start\_size=8)
- void **swap** ([hash\\_prime\\_size\\_policy](#) &other)

### Protected Member Functions

- [size\\_type](#) **get\_nearest\_larger\_size** ([size\\_type](#) size) const
- [size\\_type](#) **get\_nearest\_smaller\_size** ([size\\_type](#) size) const

#### 4.334.1 Detailed Description

A size policy whose sequence of sizes form a nearly-exponential sequence of primes.

Definition at line 450 of file hash\_policy.hpp.

#### 4.334.2 Member Typedef Documentation

##### 4.334.2.1 typedef std::size\_t \_\_gnu\_pbds::hash\_prime\_size\_policy::size\_type

Size type.

Definition at line 454 of file hash\_policy.hpp.

#### 4.334.3 Constructor & Destructor Documentation

##### 4.334.3.1 \_\_gnu\_pbds::hash\_prime\_size\_policy::hash\_prime\_size\_policy ( size\_type start\_size = 8 ) [inline]

Default constructor, or onstructor taking a start\_size The policy will use the sequence of sizes approximately start\_size, start\_size\* 2, start\_size\* 2^2, ...

Definition at line 127 of file hash\_policy.hpp.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

#### 4.335 \_\_gnu\_pbds::hash\_standard\_resize\_policy< Size\_Policy, Trigger\_Policy, External\_Size\_Access, Size\_Type > Class Template Reference

Inherits [Size\\_Policy](#), and [Trigger\\_Policy](#).

### Public Types

- enum { **external\_size\_access** }
- typedef [Size\\_Policy](#) **size\_policy**
- typedef [Size\\_Type](#) **size\_type**
- typedef [Trigger\\_Policy](#) **trigger\_policy**

### Public Member Functions

- [hash\\_standard\\_resize\\_policy](#) ()
- [hash\\_standard\\_resize\\_policy](#) (const [Size\\_Policy](#) &r\_size\_policy)
- [hash\\_standard\\_resize\\_policy](#) (const [Size\\_Policy](#) &r\_size\_policy, const [Trigger\\_Policy](#) &r\_trigger\_policy)
- [size\\_type](#) **get\_actual\_size** () const
- [Size\\_Policy](#) & **get\_size\_policy** ()

- `const Size_Policy & get_size_policy () const`
- `Trigger_Policy & get_trigger_policy ()`
- `const Trigger_Policy & get_trigger_policy () const`
- `void resize (size_type suggested_new_size)`
- `void swap (hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type > &other)`

#### Protected Member Functions

- `size_type get_new_size (size_type size, size_type num_used_e) const`
- `bool is_resize_needed () const`
- `void notify_cleared ()`
- `void notify_erase_search_collision ()`
- `void notify_erase_search_end ()`
- `void notify_erase_search_start ()`
- `void notify_erased (size_type num_e)`
- `void notify_find_search_collision ()`
- `void notify_find_search_end ()`
- `void notify_find_search_start ()`
- `void notify_insert_search_collision ()`
- `void notify_insert_search_end ()`
- `void notify_insert_search_start ()`
- `void notify_inserted (size_type num_e)`
- `void notify_resized (size_type new_size)`

#### 4.335.1 Detailed Description

```
template<typename Size_Policy = hash_exponential_size_policy<>, typename Trigger_Policy = hash_load_check_resize_↵
trigger<>, bool External_Size_Access = false, typename Size_Type = std::size_t>class __gnu_pbds::hash_standard_resize_policy<
Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >
```

A resize policy which delegates operations to size and trigger policies.

Definition at line 489 of file `hash_policy.hpp`.

#### 4.335.2 Constructor & Destructor Documentation

4.335.2.1 `template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type > __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::hash_standard_resize_policy ( )`

Default constructor.

Definition at line 44 of file `hash_policy.hpp`.

4.335.2.2 `template<typename Size_Policy, typename Trigger_Policy , bool External_Size_Access, typename Size_Type > __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::hash_standard_resize_policy ( const Size_Policy & r_size_policy )`

constructor taking some policies `r_size_policy` will be copied by the `Size_Policy` object of this object.

Definition at line 50 of file `hash_policy.hpp`.

4.335.2.3 `template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size_Type >  
 __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type  
 >::hash_standard_resize_policy ( const Size_Policy & r_size_policy, const Trigger_Policy & r_trigger_policy )`

constructor taking some policies. `r_size_policy` will be copied by the `Size_Policy` object of this object. `r_trigger_policy` will be copied by the `Trigger_Policy` object of this object.

Definition at line 56 of file `hash_policy.hpp`.

#### 4.335.3 Member Function Documentation

4.335.3.1 `template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type >  
 hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::size_type  
 __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type  
 >::get_actual_size ( ) const [inline]`

Returns the actual size of the container.

Definition at line 177 of file `hash_policy.hpp`.

4.335.3.2 `template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type >  
 hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::size_type  
 __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type  
 >::get_new_size ( size_type size, size_type num_used_e ) const [protected]`

Queries what the new size should be, when the container is resized naturally. The current `__size` of the container is `size`, and the number of used entries within the container is `num_used_e`.

Definition at line 158 of file `hash_policy.hpp`.

4.335.3.3 `template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type >  
 Size_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access,  
 Size_Type >::get_size_policy ( )`

Access to the `Size_Policy` object used.

Definition at line 242 of file `hash_policy.hpp`.

4.335.3.4 `template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type > const  
 Size_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access,  
 Size_Type >::get_size_policy ( ) const`

Const access to the `Size_Policy` object used.

Definition at line 248 of file `hash_policy.hpp`.

4.335.3.5 `template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type >  
 Trigger_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access,  
 Size_Type >::get_trigger_policy ( )`

Access to the `Trigger_Policy` object used.

Definition at line 230 of file `hash_policy.hpp`.

4.335.3.6 `template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type > const Trigger_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::get_trigger_policy ( ) const`

Access to the `Trigger_Policy` object used.

Definition at line 236 of file `hash_policy.hpp`.

4.335.3.7 `template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type > void __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::resize ( size_type suggested_new_size )`

Resizes the container to `suggested_new_size`, a suggested size (the actual size will be determined by the `Size_Policy` object).

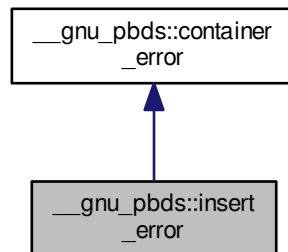
Definition at line 186 of file `hash_policy.hpp`.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

## 4.336 `__gnu_pbds::insert_error` Struct Reference

Inheritance diagram for `__gnu_pbds::insert_error`:



### 4.336.1 Detailed Description

An entry cannot be inserted into a container object for logical reasons (not, e.g., if memory is unavailable, in which case the `allocator_type`'s exception will be thrown).

Definition at line 66 of file `exception.hpp`.

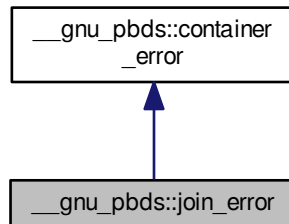
The documentation for this struct was generated from the following file:

- [exception.hpp](#)



### 4.337 `__gnu_pbds::join_error` Struct Reference

Inheritance diagram for `__gnu_pbds::join_error`:



#### 4.337.1 Detailed Description

A join cannot be performed logical reasons (i.e., the ranges of the two container objects being joined overlaps.

Definition at line 70 of file `exception.hpp`.

The documentation for this struct was generated from the following file:

- [exception.hpp](#)

### 4.338 `__gnu_pbds::linear_probe_fn< Size_Type >` Class Template Reference

#### Public Types

- typedef `Size_Type` **size\_type**

#### Public Member Functions

- void **swap** ([linear\\_probe\\_fn< Size\\_Type >](#) &other)

#### Protected Member Functions

- `size_type` [operator\(\)](#) (`size_type` i) const

#### 4.338.1 Detailed Description

```
template<typename Size_Type = std::size_t>class __gnu_pbds::linear_probe_fn< Size_Type >
```

A probe sequence policy using fixed increments.

Definition at line 61 of file `hash_policy.hpp`.

## 4.338.2 Member Function Documentation

4.338.2.1 `template<typename Size_Type > linear_probe_fn< Size_Type >::size_type __gnu_pbds::linear_probe_fn< Size_Type >::operator()( size_type i ) const` `[inline], [protected]`

Returns the i-th offset from the hash value.

Definition at line 51 of file `hash_policy.hpp`.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

4.339 `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >` Class Template Reference

Inherits type< Key, Mapped, \_Alloc, list\_update\_tag, \_\_gnu\_cxx::typelist::create2< Eq\_Fn, Update\_Policy >::type >.

## Public Types

- typedef `list_update_tag` `container_category`
- typedef `Eq_Fn` `eq_fn`
- typedef `Update_Policy` `update_policy`

## Public Member Functions

- template<typename It > `list_update` (It first, It last)
- `list_update` (const `list_update` &other)
- `list_update` & `operator=` (const `list_update` &other)
- void `swap` (`list_update` &other)

## 4.339.1 Detailed Description

`template<typename Key, typename Mapped, class Eq_Fn = typename detail::default_eq_fn<Key>::type, class Update_Policy = detail::default_update_policy::type, class _Alloc = std::allocator<char>> class __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >`

A list-update based associative container.

## Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Eq_Fn</i>	Equal functor.
<i>Update_Policy</i>	Update policy, determines when an element will be moved to the front of the list.
<i>_Alloc</i>	Allocator type.

Base is `detail::lu_map`.

Definition at line 815 of file `assoc_container.hpp`.

## 4.339.2 Constructor &amp; Destructor Documentation

4.339.2.1 `template<typename Key , typename Mapped , class Eq_Fn = typename detail::default_eq_fn<Key>::type, class Update_Policy = detail::default_update_policy::type, class _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >::list_update ( It first, It last )`  
`[inline]`

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

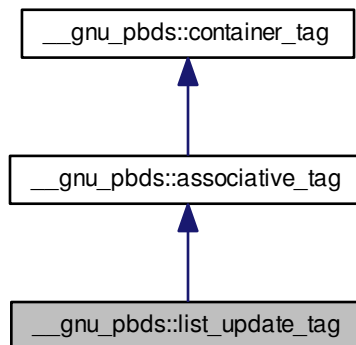
Definition at line 831 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

#### 4.340 `__gnu_pbds::list_update_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::list_update_tag`:



##### 4.340.1 Detailed Description

List-update.

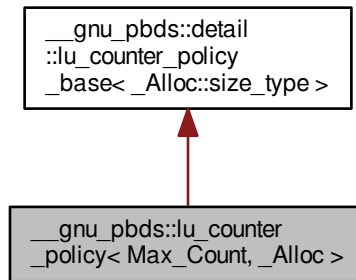
Definition at line 168 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

4.341 `__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >`:



## Public Types

- enum { `max_count` }
- typedef `_Alloc allocator_type`
- typedef `__rebind_m::other::reference metadata_reference`
- typedef `detail::lu_counter_metadata< size_type > metadata_type`
- typedef `allocator_type::size_type size_type`

## Public Member Functions

- `metadata_type operator() () const`
- `bool operator() (metadata_reference r_data) const`

## Private Member Functions

- `lu_counter_metadata< size_type > operator() (size_type max_size) const`
- `bool operator() (Metadata_Reference r_data, size_type m_max_count) const`

## 4.341.1 Detailed Description

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>> class __gnu_pbds::lu_counter_policy< Max_Count,
_Alloc >
```

A list-update policy that moves elements to the front of the list based on the counter algorithm.

Definition at line 92 of file `list_update_policy.hpp`.

#### 4.341.2 Member Typedef Documentation

4.341.2.1 `template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>> typedef __rebind_m::other::reference  
__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::metadata_reference`

Reference to metadata on which this functor operates.

Definition at line 115 of file `list_update_policy.hpp`.

4.341.2.2 `template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>> typedef  
detail::lu_counter_metadata<size_type> __gnu_pbds::lu_counter_policy< Max_Count, _Alloc  
>::metadata_type`

Metadata on which this functor operates.

Definition at line 107 of file `list_update_policy.hpp`.

#### 4.341.3 Member Enumeration Documentation

4.341.3.1 `template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>> anonymous enum`

Enumerator

**`max_count`** When some element is accessed this number of times, it will be moved to the front of the list.

Definition at line 99 of file `list_update_policy.hpp`.

#### 4.341.4 Member Function Documentation

4.341.4.1 `template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>> metadata_type  
__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::operator() ( ) const [inline]`

Creates a metadata object.

Definition at line 119 of file `list_update_policy.hpp`.

References `__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::max_count`.

4.341.4.2 `template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>> bool __gnu_pbds::  
lu_counter_policy< Max_Count, _Alloc >::operator() ( metadata_reference r_data ) const  
[inline]`

Decides whether a metadata object should be moved to the front of the list.

Definition at line 125 of file `list_update_policy.hpp`.

References `__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::max_count`.

The documentation for this class was generated from the following file:

- [list\\_update\\_policy.hpp](#)

### 4.342 `__gnu_pbds::lu_move_to_front_policy< _Alloc >` Class Template Reference

Public Types

- typedef `_Alloc` **`allocator_type`**

- typedef `__rebind_m::other::reference` [metadata\\_reference](#)
- typedef [null\\_type](#) `metadata_type`

#### Public Member Functions

- [metadata\\_type operator\(\)](#) ( ) const
- bool [operator\(\)](#) ([metadata\\_reference](#) r\_metadata) const

#### 4.342.1 Detailed Description

`template<typename _Alloc = std::allocator<char>> class __gnu_pbds::lu_move_to_front_policy<_Alloc>`

A list-update policy that unconditionally moves elements to the front of the list. A null type means that each link in a list-based container does not actually need metadata.

Definition at line 57 of file `list_update_policy.hpp`.

#### 4.342.2 Member Typedef Documentation

4.342.2.1 `template<typename _Alloc = std::allocator<char>> typedef __rebind_m::other::reference  
__gnu_pbds::lu_move_to_front_policy<_Alloc>::metadata_reference`

Reference to metadata on which this functor operates.

Definition at line 70 of file `list_update_policy.hpp`.

4.342.2.2 `template<typename _Alloc = std::allocator<char>> typedef null_type __gnu_pbds::lu_move_to_front_  
policy<_Alloc>::metadata_type`

Metadata on which this functor operates.

Definition at line 63 of file `list_update_policy.hpp`.

#### 4.342.3 Member Function Documentation

4.342.3.1 `template<typename _Alloc = std::allocator<char>> metadata_type __gnu_pbds::lu_move_to_front_policy<  
_Alloc>::operator()( ) const [inline]`

Creates a metadata object.

Definition at line 74 of file `list_update_policy.hpp`.

4.342.3.2 `template<typename _Alloc = std::allocator<char>> bool __gnu_pbds::lu_move_to_front_policy<_Alloc  
>::operator()( metadata_reference r_metadata ) const [inline]`

Decides whether a metadata object should be moved to the front of the list.

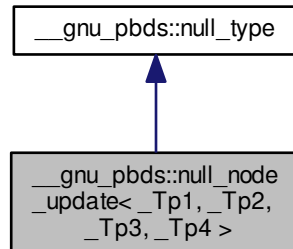
Definition at line 80 of file `list_update_policy.hpp`.

The documentation for this class was generated from the following file:

- [list\\_update\\_policy.hpp](#)

#### 4.343 `__gnu_pbds::null_node_update<_Tp1, _Tp2, _Tp3, _Tp4>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::null_node_update<_Tp1, _Tp2, _Tp3, _Tp4>`:



##### 4.343.1 Detailed Description

`template<typename _Tp1, typename _Tp2, typename _Tp3, typename _Tp4> struct __gnu_pbds::null_node_update<_Tp1, _Tp2, _Tp3, _Tp4>`

A null node updater, indicating that no node updates are required.

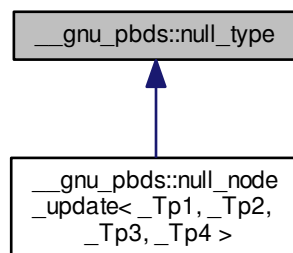
Definition at line 214 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.344 `__gnu_pbds::null_type` Struct Reference

Inheritance diagram for `__gnu_pbds::null_type`:



## 4.344.1 Detailed Description

Represents no type, or absence of type, for template tricks.

In a mapped-policy, indicates that an associative container is a set.

In a list-update policy, indicates that each link does not need metadata.

In a hash policy, indicates that the combining hash function is actually a ranged hash function.

In a probe policy, indicates that the combining probe function is actually a ranged probe function.

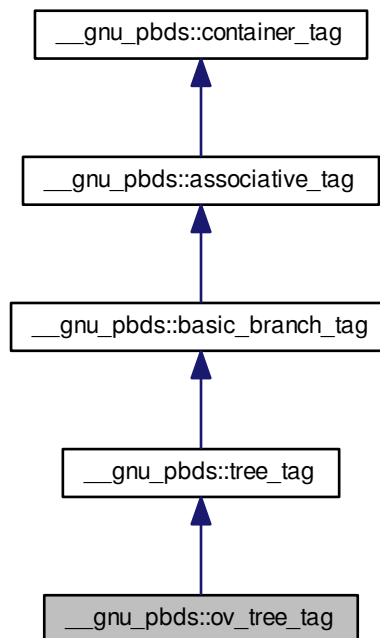
Definition at line 210 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.345 \_\_gnu\_pbds::ov\_tree\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::ov\_tree\_tag:



## 4.345.1 Detailed Description

Ordered-vector tree.

Definition at line 159 of file tag\_and\_trait.hpp.

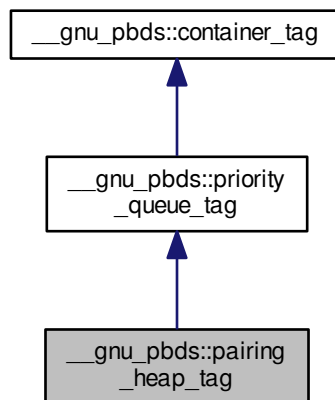


The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.346 `__gnu_pbds::pairing_heap_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::pairing_heap_tag`:



##### 4.346.1 Detailed Description

Pairing-heap.

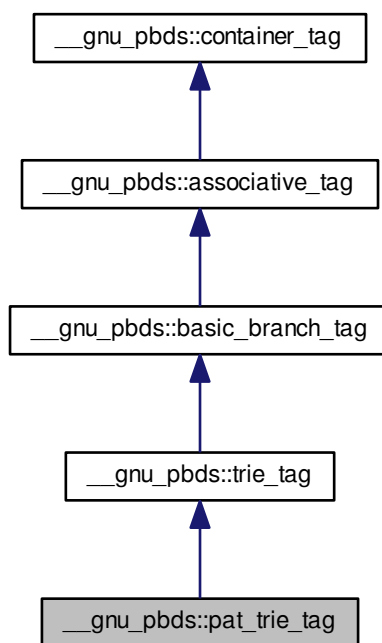
Definition at line 174 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.347 \_\_gnu\_pbds::pat\_trie\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::pat\_trie\_tag:



## 4.347.1 Detailed Description

PATRICIA trie.

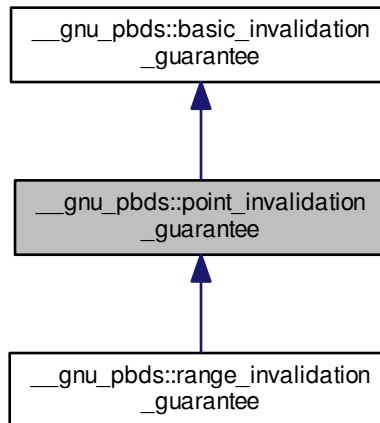
Definition at line 165 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.348 `__gnu_pbds::point_invalidation_guarantee` Struct Reference

Inheritance diagram for `__gnu_pbds::point_invalidation_guarantee`:



##### 4.348.1 Detailed Description

Signifies an invalidation guarantee that includes all those of its base, and additionally, that any point-type iterator, pointer, or reference to a container object's mapped value type is valid as long as its corresponding entry has not be erased, regardless of modifications to the container object.

Definition at line 103 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.349 `__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>` Class Template Reference

Inherits `type<_Tv, Cmp_Fn, _Alloc, Tag>`.

##### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `__rebind_va::const_pointer` **const\_pointer**
- typedef `__rebind_va::const_reference` **const\_reference**
- typedef `Tag` **container\_category**
- typedef `allocator_type::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**

- `typedef base_type::point_const_iterator` **point\_const\_iterator**
- `typedef base_type::point_iterator` **point\_iterator**
- `typedef __rebind_va::pointer` **pointer**
- `typedef __rebind_va::reference` **reference**
- `typedef allocator_type::size_type` **size\_type**
- `typedef _Tv` **value\_type**

#### Public Member Functions

- [priority\\_queue](#) (const `cmp_fn` &`r_cmp_fn`)
- `template<typename It>` [priority\\_queue](#) (It `first_it`, It `last_it`)
- `template<typename It>` [priority\\_queue](#) (It `first_it`, It `last_it`, const `cmp_fn` &`r_cmp_fn`)
- **priority\_queue** (const [priority\\_queue](#) &`other`)
- [priority\\_queue](#) & **operator=** (const [priority\\_queue](#) &`other`)
- void **swap** ([priority\\_queue](#) &`other`)

#### 4.349.1 Detailed Description

```
template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename _Alloc = std::
::allocator<char>> class __gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>
```

A priority queue composed of one specific heap policy.

#### Template Parameters

<code>_Tv</code>	Value type.
<code>Cmp_Fn</code>	Comparison functor.
<code>Tag</code>	Instantiating data structure type, see <code>container_tag</code> .
<code>_Alloc</code>	Allocator type.

Base is dispatched at compile time via `Tag`, from the following choices: `binary_heap_tag`, `binomial_heap_tag`, `pairing_heap_tag`, `rc_binomial_heap_tag`, `thin_heap_tag`

Base choices are: `detail::binary_heap`, `detail::binomial_heap`, `detail::pairing_heap`, `detail::rc_binomial_heap`, `detail::thin_heap`.

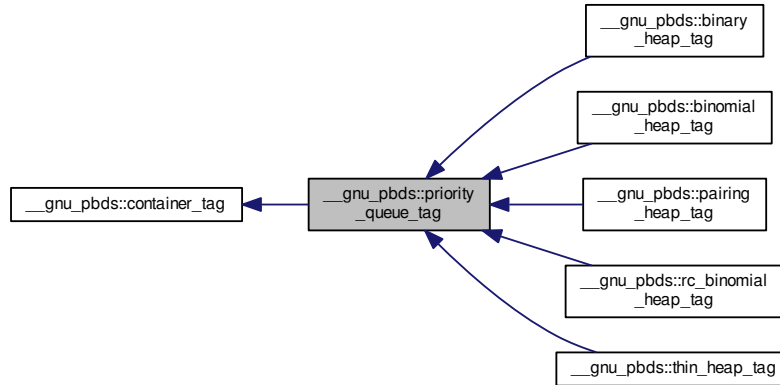
Definition at line 83 of file `priority_queue.hpp`.

The documentation for this class was generated from the following file:

- [priority\\_queue.hpp](#)

### 4.350 `__gnu_pbds::priority_queue_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::priority_queue_tag`:



#### 4.350.1 Detailed Description

Basic priority-queue.

Definition at line 171 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 4.351 `__gnu_pbds::quadratic_probe_fn< Size_Type >` Class Template Reference

#### Public Types

- typedef `Size_Type` **size\_type**

#### Public Member Functions

- void **swap** ([quadratic\\_probe\\_fn< Size\\_Type >](#) &other)

#### Protected Member Functions

- `size_type` [operator\(\)](#) (`size_type` i) const

#### 4.351.1 Detailed Description

```
template<typename Size_Type = std::size_t>class __gnu_pbds::quadratic_probe_fn< Size_Type >
```

A probe sequence policy using square increments.

Definition at line 85 of file `hash_policy.hpp`.

#### 4.351.2 Member Function Documentation

4.351.2.1 `template<typename Size_Type > quadratic_probe_fn< Size_Type >::size_type __gnu_pbds::quadratic_probe_fn< Size_Type >::operator() ( size_type i ) const` `[inline]`, `[protected]`

Returns the i-th offset from the hash value.

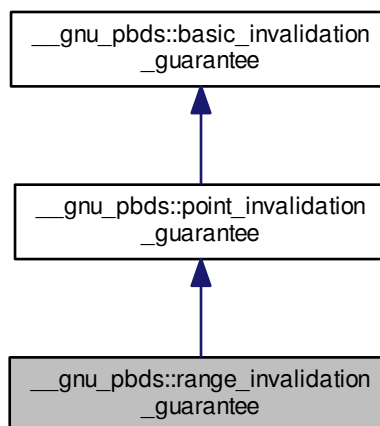
Definition at line 51 of file `hash_policy.hpp`.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

#### 4.352 `__gnu_pbds::range_invalidation_guarantee` Struct Reference

Inheritance diagram for `__gnu_pbds::range_invalidation_guarantee`:



#### 4.352.1 Detailed Description

Signifies an invalidation guarantee that includes all those of its base, and additionally, that any range-type iterator (including the returns of `begin()` and `end()`) is in the correct relative positions to other range-type iterators as long as its corresponding entry has not been erased, regardless of modifications to the container object.

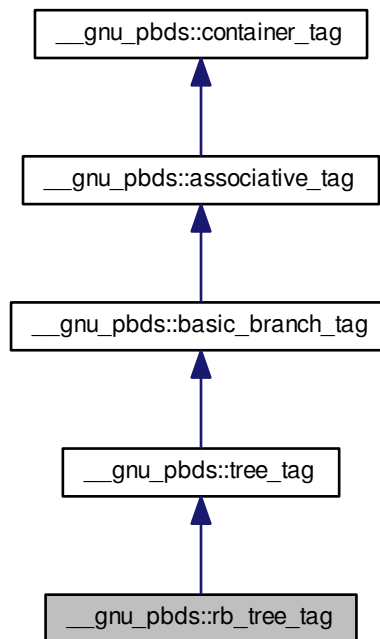
Definition at line 114 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 4.353 \_\_gnu\_pbds::rb\_tree\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::rb\_tree\_tag:



#### 4.353.1 Detailed Description

Red-black tree.

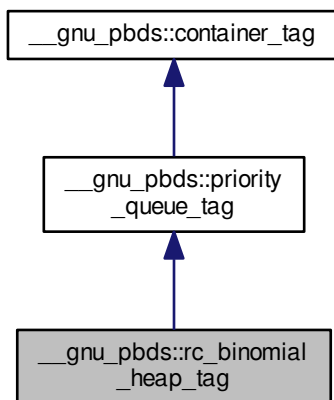
Definition at line 153 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.354 \_\_gnu\_pbds::rc\_binomial\_heap\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::rc\_binomial\_heap\_tag:



## 4.354.1 Detailed Description

Redundant-counter binomial-heap.

Definition at line 180 of file `tag_and_trait.hpp`.

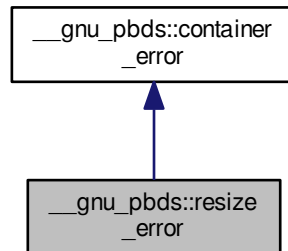
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)



### 4.355 `__gnu_pbds::resize_error` Struct Reference

Inheritance diagram for `__gnu_pbds::resize_error`:



#### 4.355.1 Detailed Description

A container cannot be resized.

Definition at line 73 of file `exception.hpp`.

The documentation for this struct was generated from the following file:

- [exception.hpp](#)

### 4.356 `__gnu_pbds::sample_probe_fn` Class Reference

#### Public Types

- typedef `std::size_t` **size\_type**

#### Public Member Functions

- [sample\\_probe\\_fn](#) ()
- [sample\\_probe\\_fn](#) (const [sample\\_probe\\_fn](#) &)
- void [swap](#) ([sample\\_probe\\_fn](#) &)

#### Protected Member Functions

- `size_type` [operator\(\)](#) (key\_const\_reference r\_key, size\_type i) const

#### 4.356.1 Detailed Description

A sample probe policy.

Definition at line 47 of file `sample_probe_fn.hpp`.

## 4.356.2 Constructor &amp; Destructor Documentation

4.356.2.1 `__gnu_pbds::sample_probe_fn::sample_probe_fn ( )`

Default constructor.

4.356.2.2 `__gnu_pbds::sample_probe_fn::sample_probe_fn ( const sample_probe_fn & )`

Copy constructor.

## 4.356.3 Member Function Documentation

4.356.3.1 `size_type __gnu_pbds::sample_probe_fn::operator() ( key_const_reference r_key, size_type i ) const` `[inline]`, `[protected]`

Returns the *i*-th offset from the hash value of some key *r\_key*.

4.356.3.2 `void __gnu_pbds::sample_probe_fn::swap ( sample_probe_fn & )` `[inline]`

Swaps content.

The documentation for this class was generated from the following file:

- [sample\\_probe\\_fn.hpp](#)

4.357 `__gnu_pbds::sample_range_hashing` Class Reference

## Public Types

- `typedef std::size_t` [size\\_type](#)

## Public Member Functions

- [sample\\_range\\_hashing](#) ( )
- [sample\\_range\\_hashing](#) (const [sample\\_range\\_hashing](#) &other)
- void [swap](#) ([sample\\_range\\_hashing](#) &other)

## Protected Member Functions

- void [notify\\_resized](#) ([size\\_type](#))
- [size\\_type](#) [operator\(\)](#) ([size\\_type](#)) const

## 4.357.1 Detailed Description

A sample range-hashing functor.

Definition at line 47 of file `sample_range_hashing.hpp`.

#### 4.357.2 Member Typedef Documentation

##### 4.357.2.1 `typedef std::size_t __gnu_pbds::sample_range_hashing::size_type`

Size type.

Definition at line 51 of file `sample_range_hashing.hpp`.

#### 4.357.3 Constructor & Destructor Documentation

##### 4.357.3.1 `__gnu_pbds::sample_range_hashing::sample_range_hashing ( )`

Default constructor.

##### 4.357.3.2 `__gnu_pbds::sample_range_hashing::sample_range_hashing ( const sample_range_hashing & other )`

Copy constructor.

#### 4.357.4 Member Function Documentation

##### 4.357.4.1 `void __gnu_pbds::sample_range_hashing::notify_resized ( size_type )` `[protected]`

Notifies the policy object that the container's size has changed to argument's size.

##### 4.357.4.2 `size_type __gnu_pbds::sample_range_hashing::operator() ( size_type ) const` `[inline]`, `[protected]`

Transforms the `__hash` value hash into a ranged-hash value.

##### 4.357.4.3 `void __gnu_pbds::sample_range_hashing::swap ( sample_range_hashing & other )` `[inline]`

Swaps content.

The documentation for this class was generated from the following file:

- [sample\\_range\\_hashing.hpp](#)

### 4.358 `__gnu_pbds::sample_ranged_hash_fn` Class Reference

#### Public Types

- `typedef std::size_t size_type`

#### Public Member Functions

- [sample\\_ranged\\_hash\\_fn \(\)](#)
- [sample\\_ranged\\_hash\\_fn \(const sample\\_ranged\\_hash\\_fn &\)](#)
- `void swap (sample_ranged_hash_fn &)`

#### Protected Member Functions

- `void notify_resized (size_type)`
- `size_type operator() (key_const_reference) const`

## 4.358.1 Detailed Description

A sample ranged-hash functor.

Definition at line 47 of file `sample_ranged_hash_fn.hpp`.

## 4.358.2 Constructor &amp; Destructor Documentation

4.358.2.1 `__gnu_pbds::sample_ranged_hash_fn::sample_ranged_hash_fn ( )`

Default constructor.

4.358.2.2 `__gnu_pbds::sample_ranged_hash_fn::sample_ranged_hash_fn ( const sample_ranged_hash_fn & )`

Copy constructor.

## 4.358.3 Member Function Documentation

4.358.3.1 `void __gnu_pbds::sample_ranged_hash_fn::notify_resized ( size_type )` `[protected]`

Notifies the policy object that the container's `__size` has changed to `size`.

4.358.3.2 `size_type __gnu_pbds::sample_ranged_hash_fn::operator() ( key_const_reference ) const` `[inline]`, `[protected]`

Transforms `key_const_reference` into a position within the table.

4.358.3.3 `void __gnu_pbds::sample_ranged_hash_fn::swap ( sample_ranged_hash_fn & )` `[inline]`

Swaps content.

The documentation for this class was generated from the following file:

- [sample\\_ranged\\_hash\\_fn.hpp](#)

4.359 `__gnu_pbds::sample_ranged_probe_fn` Class Reference

## Public Types

- typedef `std::size_t` **size\_type**

## Public Member Functions

- **sample\_ranged\_probe\_fn** (const [sample\\_ranged\\_probe\\_fn](#) &)
- void **swap** ([sample\\_ranged\\_probe\\_fn](#) &)

## Protected Member Functions

- void **notify\_resized** (size\_type)
- size\_type **operator()** (key\_const\_reference, std::size\_t, size\_type) const

#### 4.359.1 Detailed Description

A sample ranged-probe functor.

Definition at line 47 of file `sample_ranged_probe_fn.hpp`.

The documentation for this class was generated from the following file:

- [sample\\_ranged\\_probe\\_fn.hpp](#)

### 4.360 `__gnu_pbds::sample_resize_policy` Class Reference

#### Public Types

- typedef std::size\_t [size\\_type](#)

#### Public Member Functions

- [sample\\_resize\\_policy](#) ()
- [sample\\_range\\_hashing](#) (const [sample\\_resize\\_policy](#) &other)
- void [swap](#) ([sample\\_resize\\_policy](#) &other)

#### Protected Member Functions

- [size\\_type](#) [get\\_new\\_size](#) ([size\\_type](#) size, [size\\_type](#) num\_used\_e) const
- bool [is\\_resize\\_needed](#) () const
- void [notify\\_cleared](#) ()
- void [notify\\_erase\\_search\\_collision](#) ()
- void [notify\\_erase\\_search\\_end](#) ()
- void [notify\\_erase\\_search\\_start](#) ()
- void [notify\\_erased](#) ([size\\_type](#) num\_e)
- void [notify\\_find\\_search\\_collision](#) ()
- void [notify\\_find\\_search\\_end](#) ()
- void [notify\\_find\\_search\\_start](#) ()
- void [notify\\_insert\\_search\\_collision](#) ()
- void [notify\\_insert\\_search\\_end](#) ()
- void [notify\\_insert\\_search\\_start](#) ()
- void [notify\\_inserted](#) ([size\\_type](#) num\_e)
- void [notify\\_resized](#) ([size\\_type](#) new\_size)

#### 4.360.1 Detailed Description

A sample resize policy.

Definition at line 47 of file `sample_resize_policy.hpp`.

#### 4.360.2 Member Typedef Documentation

##### 4.360.2.1 typedef std::size\_t `__gnu_pbds::sample_resize_policy::size_type`

Size type.

Definition at line 51 of file `sample_resize_policy.hpp`.

### 4.360.3 Constructor & Destructor Documentation

#### 4.360.3.1 \_\_gnu\_pbds::sample\_resize\_policy::sample\_resize\_policy ( )

Default constructor.

### 4.360.4 Member Function Documentation

#### 4.360.4.1 size\_type \_\_gnu\_pbds::sample\_resize\_policy::get\_new\_size ( size\_type *size*, size\_type *num\_used\_e* ) const [protected]

Queries what the new size should be.

#### 4.360.4.2 bool \_\_gnu\_pbds::sample\_resize\_policy::is\_resize\_needed ( ) const [inline], [protected]

Queries whether a resize is needed.

#### 4.360.4.3 void \_\_gnu\_pbds::sample\_resize\_policy::notify\_cleared ( ) [protected]

Notifies the table was cleared.

#### 4.360.4.4 void \_\_gnu\_pbds::sample\_resize\_policy::notify\_erase\_search\_collision ( ) [inline], [protected]

Notifies a search encountered a collision.

#### 4.360.4.5 void \_\_gnu\_pbds::sample\_resize\_policy::notify\_erase\_search\_end ( ) [inline], [protected]

Notifies a search ended.

#### 4.360.4.6 void \_\_gnu\_pbds::sample\_resize\_policy::notify\_erase\_search\_start ( ) [inline], [protected]

Notifies a search started.

#### 4.360.4.7 void \_\_gnu\_pbds::sample\_resize\_policy::notify\_erased ( size\_type *num\_e* ) [inline], [protected]

Notifies an element was erased.

#### 4.360.4.8 void \_\_gnu\_pbds::sample\_resize\_policy::notify\_find\_search\_collision ( ) [inline], [protected]

Notifies a search encountered a collision.

#### 4.360.4.9 void \_\_gnu\_pbds::sample\_resize\_policy::notify\_find\_search\_end ( ) [inline], [protected]

Notifies a search ended.

#### 4.360.4.10 void \_\_gnu\_pbds::sample\_resize\_policy::notify\_find\_search\_start ( ) [inline], [protected]

Notifies a search started.

#### 4.360.4.11 void \_\_gnu\_pbds::sample\_resize\_policy::notify\_insert\_search\_collision ( ) [inline], [protected]

Notifies a search encountered a collision.

#### 4.360.4.12 void \_\_gnu\_pbds::sample\_resize\_policy::notify\_insert\_search\_end ( ) [inline], [protected]

Notifies a search ended.

4.360.4.13 void `__gnu_pbds::sample_resize_policy::notify_insert_search_start ( )` `[inline]`, `[protected]`

Notifies a search started.

4.360.4.14 void `__gnu_pbds::sample_resize_policy::notify_inserted ( size_type num_e )` `[inline]`, `[protected]`

Notifies an element was inserted.

4.360.4.15 void `__gnu_pbds::sample_resize_policy::notify_resized ( size_type new_size )` `[protected]`

Notifies the table was resized to new\_size.

4.360.4.16 `__gnu_pbds::sample_resize_policy::sample_range_hashing ( const sample_resize_policy & other )`

Copy constructor.

4.360.4.17 void `__gnu_pbds::sample_resize_policy::swap ( sample_resize_policy & other )` `[inline]`

Swaps content.

The documentation for this class was generated from the following file:

- [sample\\_resize\\_policy.hpp](#)

## 4.361 `__gnu_pbds::sample_resize_trigger` Class Reference

### Public Types

- typedef std::size\_t [size\\_type](#)

### Public Member Functions

- [sample\\_resize\\_trigger \(\)](#)
- [sample\\_range\\_hashing](#) (const [sample\\_resize\\_trigger](#) &)
- void [swap](#) ([sample\\_resize\\_trigger](#) &)

### Protected Member Functions

- bool [is\\_grow\\_needed](#) ([size\\_type](#) size, [size\\_type](#) num\_entries) const
- bool [is\\_resize\\_needed](#) () const
- void [notify\\_cleared](#) ()
- void [notify\\_erase\\_search\\_collision](#) ()
- void [notify\\_erase\\_search\\_end](#) ()
- void [notify\\_erase\\_search\\_start](#) ()
- void [notify\\_erased](#) ([size\\_type](#) num\_entries)
- void [notify\\_externally\\_resized](#) ([size\\_type](#) new\_size)
- void [notify\\_find\\_search\\_collision](#) ()
- void [notify\\_find\\_search\\_end](#) ()
- void [notify\\_find\\_search\\_start](#) ()
- void [notify\\_insert\\_search\\_collision](#) ()
- void [notify\\_insert\\_search\\_end](#) ()
- void [notify\\_insert\\_search\\_start](#) ()
- void [notify\\_inserted](#) ([size\\_type](#) num\_entries)
- void [notify\\_resized](#) ([size\\_type](#) new\_size)

#### 4.361.1 Detailed Description

A sample resize trigger policy.

Definition at line 47 of file sample\_resize\_trigger.hpp.

#### 4.361.2 Member Typedef Documentation

##### 4.361.2.1 typedef std::size\_t \_\_gnu\_pbds::sample\_resize\_trigger::size\_type

Size type.

Definition at line 51 of file sample\_resize\_trigger.hpp.

#### 4.361.3 Constructor & Destructor Documentation

##### 4.361.3.1 \_\_gnu\_pbds::sample\_resize\_trigger::sample\_resize\_trigger ( )

Default constructor.

#### 4.361.4 Member Function Documentation

##### 4.361.4.1 bool \_\_gnu\_pbds::sample\_resize\_trigger::is\_grow\_needed ( size\_type size, size\_type num\_entries ) const [inline], [protected]

Queries whether a grow is needed.

##### 4.361.4.2 bool \_\_gnu\_pbds::sample\_resize\_trigger::is\_resize\_needed ( ) const [inline], [protected]

Queries whether a resize is needed.

##### 4.361.4.3 void \_\_gnu\_pbds::sample\_resize\_trigger::notify\_cleared ( ) [protected]

Notifies the table was cleared.

##### 4.361.4.4 void \_\_gnu\_pbds::sample\_resize\_trigger::notify\_erase\_search\_collision ( ) [inline], [protected]

Notifies a search encountered a collision.

##### 4.361.4.5 void \_\_gnu\_pbds::sample\_resize\_trigger::notify\_erase\_search\_end ( ) [inline], [protected]

Notifies a search ended.

##### 4.361.4.6 void \_\_gnu\_pbds::sample\_resize\_trigger::notify\_erase\_search\_start ( ) [inline], [protected]

Notifies a search started.

##### 4.361.4.7 void \_\_gnu\_pbds::sample\_resize\_trigger::notify\_erased ( size\_type num\_entries ) [inline], [protected]

Notifies an element was erased.

##### 4.361.4.8 void \_\_gnu\_pbds::sample\_resize\_trigger::notify\_externally\_resized ( size\_type new\_size ) [protected]

Notifies the table was resized externally.



4.361.4.9 void `__gnu_pbds::sample_resize_trigger::notify_find_search_collision ( )` [inline],[protected]

Notifies a search encountered a collision.

4.361.4.10 void `__gnu_pbds::sample_resize_trigger::notify_find_search_end ( )` [inline],[protected]

Notifies a search ended.

4.361.4.11 void `__gnu_pbds::sample_resize_trigger::notify_find_search_start ( )` [inline],[protected]

Notifies a search started.

4.361.4.12 void `__gnu_pbds::sample_resize_trigger::notify_insert_search_collision ( )` [inline],[protected]

Notifies a search encountered a collision.

4.361.4.13 void `__gnu_pbds::sample_resize_trigger::notify_insert_search_end ( )` [inline],[protected]

Notifies a search ended.

4.361.4.14 void `__gnu_pbds::sample_resize_trigger::notify_insert_search_start ( )` [inline],[protected]

Notifies a search started.

4.361.4.15 void `__gnu_pbds::sample_resize_trigger::notify_inserted ( size_type num_entries )` [inline],[protected]

Notifies an element was inserted. the total number of entries in the table is num\_entries.

4.361.4.16 void `__gnu_pbds::sample_resize_trigger::notify_resized ( size_type new_size )` [protected]

Notifies the table was resized as a result of this object's signifying that a resize is needed.

4.361.4.17 `__gnu_pbds::sample_resize_trigger::sample_range_hashing ( const sample_resize_trigger & )`

Copy constructor.

4.361.4.18 void `__gnu_pbds::sample_resize_trigger::swap ( sample_resize_trigger & )` [inline]

Swaps content.

The documentation for this class was generated from the following file:

- [sample\\_resize\\_trigger.hpp](#)

## 4.362 `__gnu_pbds::sample_size_policy` Class Reference

### Public Types

- typedef std::size\_t [size\\_type](#)

### Public Member Functions

- [sample\\_size\\_policy \( \)](#)
- [sample\\_range\\_hashing](#) (const [sample\\_size\\_policy](#) &)
- void [swap](#) ([sample\\_size\\_policy](#) &other)

#### Protected Member Functions

- [size\\_type get\\_nearest\\_larger\\_size \(size\\_type size\) const](#)
- [size\\_type get\\_nearest\\_smaller\\_size \(size\\_type size\) const](#)

#### 4.362.1 Detailed Description

A sample size policy.

Definition at line 47 of file `sample_size_policy.hpp`.

#### 4.362.2 Member Typedef Documentation

##### 4.362.2.1 `typedef std::size_t __gnu_pbds::sample_size_policy::size_type`

Size type.

Definition at line 51 of file `sample_size_policy.hpp`.

#### 4.362.3 Constructor & Destructor Documentation

##### 4.362.3.1 `__gnu_pbds::sample_size_policy::sample_size_policy ( )`

Default constructor.

#### 4.362.4 Member Function Documentation

##### 4.362.4.1 `size_type __gnu_pbds::sample_size_policy::get_nearest_larger_size ( size_type size ) const` `[inline]`, `[protected]`

Given a `__size` `size`, returns a `__size` that is larger.

##### 4.362.4.2 `size_type __gnu_pbds::sample_size_policy::get_nearest_smaller_size ( size_type size ) const` `[inline]`, `[protected]`

Given a `__size` `size`, returns a `__size` that is smaller.

##### 4.362.4.3 `__gnu_pbds::sample_size_policy::sample_range_hashing ( const sample_size_policy & )`

Copy constructor.

##### 4.362.4.4 `void __gnu_pbds::sample_size_policy::swap ( sample_size_policy & other )` `[inline]`

Swaps content.

The documentation for this class was generated from the following file:

- [sample\\_size\\_policy.hpp](#)

#### 4.363 `__gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc >` Class Template Reference

#### 4.363.1 Detailed Description

```
template<typename Const_Node_Iter, typename Node_Iter, typename Cmp_Fn, typename _Alloc>class __gnu_pbds::sample_tree_↵
node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc >
```

A sample node updatator.

Definition at line 49 of file sample\_tree\_node\_update.hpp.

The documentation for this class was generated from the following file:

- [sample\\_tree\\_node\\_update.hpp](#)

#### 4.364 \_\_gnu\_pbds::sample\_trie\_access\_traits Struct Reference

##### Public Types

- enum { **max\_size** }
- typedef \_Alloc::template rebind< [key\\_type](#) > **\_\_rebind\_k**
- typedef std::string::const\_iterator **const\_iterator**
- typedef char [e\\_type](#)
- typedef \_\_rebind\_k::other::const\_reference **key\_const\_reference**
- typedef [std::string](#) **key\_type**
- typedef std::size\_t **size\_type**

##### Static Public Member Functions

- static const\_iterator [begin](#) (key\_const\_reference)
- static size\_type [e\\_pos](#) ([e\\_type](#))
- static const\_iterator [end](#) (key\_const\_reference)

#### 4.364.1 Detailed Description

A sample trie element access traits.

Definition at line 47 of file sample\_trie\_access\_traits.hpp.

#### 4.364.2 Member Typedef Documentation

##### 4.364.2.1 typedef char \_\_gnu\_pbds::sample\_trie\_access\_traits::e\_type

Element type.

Definition at line 57 of file sample\_trie\_access\_traits.hpp.

#### 4.364.3 Member Function Documentation

##### 4.364.3.1 static const\_iterator \_\_gnu\_pbds::sample\_trie\_access\_traits::begin ( key\_const\_reference ) [inline], [static]

Returns a const\_iterator to the first element of r\_key.

4.364.3.2 `static size_type __gnu_pbds::sample_trie_access_traits::e_pos( e_type )` `[inline],[static]`

Maps an element to a position.

4.364.3.3 `static const_iterator __gnu_pbds::sample_trie_access_traits::end( key_const_reference )` `[inline],[static]`

Returns a `const_iterator` to the after-last element of `r_key`.

The documentation for this struct was generated from the following file:

- [sample\\_trie\\_access\\_traits.hpp](#)

## 4.365 `__gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >` Class Template Reference

### Public Types

- `typedef std::size_t metadata_type`

### Protected Member Functions

- [sample\\_trie\\_node\\_update\(\)](#)
- `void operator()` (`node_iterator`, `node_const_iterator`) `const`

### 4.365.1 Detailed Description

```
template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>class __gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >
```

A sample node updator.

Definition at line 49 of file `sample_trie_node_update.hpp`.

### 4.365.2 Constructor & Destructor Documentation

```
4.365.2.1 template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc >
__gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::sample_trie_node_update
( ) [protected]
```

Default constructor.

### 4.365.3 Member Function Documentation

```
4.365.3.1 template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc > void
__gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::operator() ( node_iterator ,
node_const_iterator ) const [inline],[protected]
```

Updates the rank of a node through a `node_iterator` `node_it`; `end_nd_it` is the end node iterator.

The documentation for this class was generated from the following file:

- [sample\\_trie\\_node\\_update.hpp](#)

## 4.366 `__gnu_pbds::sample_update_policy` Struct Reference

### Public Member Functions

- [sample\\_update\\_policy](#) ()
- [sample\\_update\\_policy](#) (const [sample\\_update\\_policy](#) &)
- void [swap](#) ([sample\\_update\\_policy](#) &other)

### Protected Types

- typedef some\_metadata\_type [metadata\\_type](#)

### Protected Member Functions

- [metadata\\_type operator\(\)](#) () const
- bool [operator\(\)](#) (metadata\_reference) const

#### 4.366.1 Detailed Description

A sample list-update policy.

Definition at line 47 of file `sample_update_policy.hpp`.

#### 4.366.2 Member Typedef Documentation

4.366.2.1 `typedef some_metadata_type __gnu_pbds::sample_update_policy::metadata_type` `[protected]`

Metadata on which this functor operates.

Definition at line 61 of file `sample_update_policy.hpp`.

#### 4.366.3 Constructor & Destructor Documentation

4.366.3.1 `__gnu_pbds::sample_update_policy::sample_update_policy ( )`

Default constructor.

4.366.3.2 `__gnu_pbds::sample_update_policy::sample_update_policy ( const sample_update_policy & )`

Copy constructor.

#### 4.366.4 Member Function Documentation

4.366.4.1 `metadata_type __gnu_pbds::sample_update_policy::operator() ( ) const` `[protected]`

Creates a metadata object.

4.366.4.2 `bool __gnu_pbds::sample_update_policy::operator() ( metadata_reference ) const` `[protected]`

Decides whether a metadata object should be moved to the front of the list. A list-update based containers object will call this method to decide whether to move a node to the front of the list. The method should return true if the node should be moved to the front of the list.

4.366.4.3 void \_\_gnu\_pbds::sample\_update\_policy::swap ( sample\_update\_policy & *other* ) [inline]

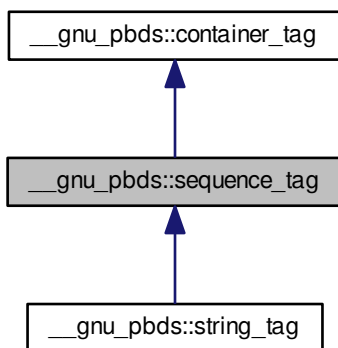
Swaps content.

The documentation for this struct was generated from the following file:

- [sample\\_update\\_policy.hpp](#)

## 4.367 \_\_gnu\_pbds::sequence\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::sequence\_tag:



### 4.367.1 Detailed Description

Basic sequence.

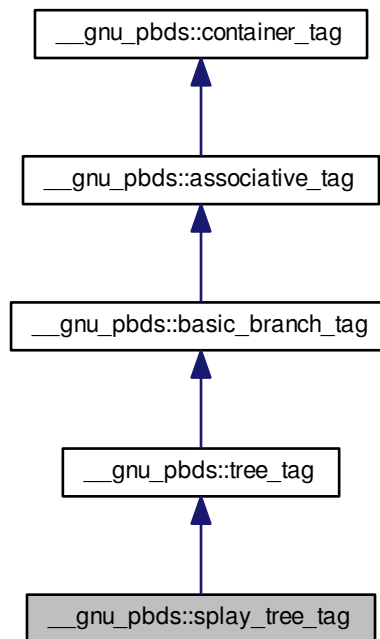
Definition at line 129 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.368 \_\_gnu\_pbds::splay\_tree\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::splay\_tree\_tag:



##### 4.368.1 Detailed Description

Splay tree.

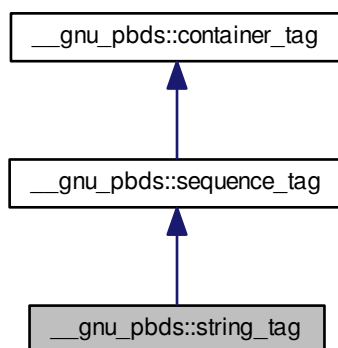
Definition at line 156 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

4.369 `__gnu_pbds::string_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::string_tag`:



## 4.369.1 Detailed Description

Basic string container, inclusive of strings, ropes, etc.

Definition at line 132 of file `tag_and_trait.hpp`.

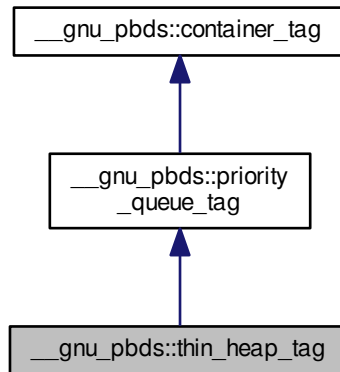
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)



#### 4.370 `__gnu_pbds::thin_heap_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::thin_heap_tag`:



##### 4.370.1 Detailed Description

Thin heap.

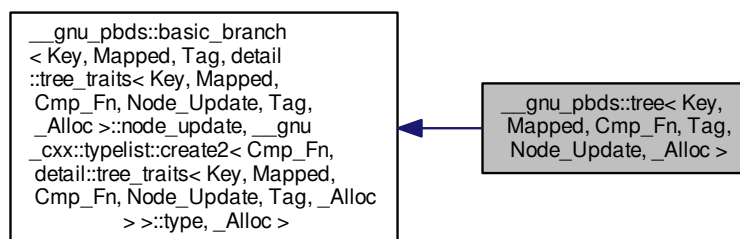
Definition at line 186 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.371 `__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >`:



## Public Types

- typedef `Cmp_Fn` `cmp_fn`
- typedef `detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Tag, _Alloc >::node_update` `node_update`

## Public Member Functions

- `tree` (`const cmp_fn &c`)
- `template<typename It > tree` (`It first, It last`)
- `template<typename It > tree` (`It first, It last, const cmp_fn &c`)
- `tree` (`const tree &other`)
- `tree & operator=` (`const tree &other`)
- `void swap` (`tree &other`)

## 4.371.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn = std::less<Key>, typename Tag = rb_tree_tag, template< typename
Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc =
std::allocator<char>> class __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >
```

A tree-based container.

## Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Cmp_Fn</i>	Comparison functor.
<i>Tag</i>	Instantiating data structure type, see <code>container_tag</code> .
<i>Node_Update</i>	Updates tree internal-nodes, restores invariants when invalidated. XXX See <code>design::tree-based-containersnode</code> invariants.
<i>_Alloc</i>	Allocator type.

Base tag choices are: `ov_tree_tag`, `rb_tree_tag`, `splay_tree_tag`.

Base is `basic_branch`.

Definition at line 635 of file `assoc_container.hpp`.

## 4.371.2 Member Typedef Documentation

4.371.2.1 `template<typename Key , typename Mapped , typename Cmp_Fn = std::less<Key>, typename Tag = rb_tree_tag, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>> typedef Cmp_Fn __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::cmp_fn`

Comparison functor type.

Definition at line 642 of file `assoc_container.hpp`.

## 4.371.3 Constructor &amp; Destructor Documentation

4.371.3.1 `template<typename Key , typename Mapped , typename Cmp_Fn = std::less<Key>, typename Tag = rb_tree_tag, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>> __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree ( const cmp_fn & c ) [inline]`

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

Definition at line 648 of file `assoc_container.hpp`.

4.371.3.2 `template<typename Key , typename Mapped , typename Cmp_Fn = std::less<Key>, typename Tag = rb_tree_tag, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree ( It first, It last ) [inline]`

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

Definition at line 655 of file `assoc_container.hpp`.

4.371.3.3 `template<typename Key , typename Mapped , typename Cmp_Fn = std::less<Key>, typename Tag = rb_tree_tag, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree ( It first, It last, const cmp_fn & c ) [inline]`

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

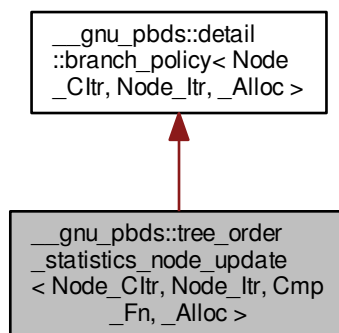
Definition at line 663 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

#### 4.372 `__gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >`:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `node_const_iterator::value_type` **const\_iterator**
- typedef `node_iterator::value_type` **iterator**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_type` **key\_type**
- typedef `size_type` **metadata\_type**
- typedef `Node_Cltr` **node\_const\_iterator**
- typedef `Node_Itr` **node\_iterator**
- typedef `allocator_type::size_type` **size\_type**

## Public Member Functions

- `const_iterator` [find\\_by\\_order](#) (`size_type`) const
- `iterator` [find\\_by\\_order](#) (`size_type`)
- `size_type` [order\\_of\\_key](#) (`key_const_reference`) const

## Protected Member Functions

- void [operator\(\)](#) (`node_iterator`, `node_const_iterator`) const

## Private Types

- typedef `Node_Itr::value_type` **it\_type**
- typedef `remove_const< key_type >::type` **rkkey\_type**
- typedef `remove_const< value_type >::type` **rcvalue\_type**
- typedef `_Alloc::template rebind< rkkey_type >::other` **rebind\_k**
- typedef `_Alloc::template rebind< rcvalue_type >::other` **rebind\_v**
- typedef `rebind_v::reference` **reference**
- typedef `std::iterator_traits< it_type >::value_type` **value\_type**

## Private Member Functions

- virtual `it_type` **end** ()=0
- `it_type` **end\_iterator** () const

## Static Private Member Functions

- static `key_const_reference` **extract\_key** (`const_reference` r\_val)

### 4.372.1 Detailed Description

`template<typename Node_Cltr, typename Node_Itr, typename Cmp_Fn, typename _Alloc>class __gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >`

Functor updating ranks of entrees.

Definition at line 64 of file `tree_policy.hpp`.

## 4.372.2 Member Function Documentation

4.372.2.1 `template<typename Node_Cltr , typename Node_Itr , typename Cmp_Fn , typename _Alloc >  
 tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >::const_iterator  
 __gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >::find_by_order (`  
`size_type order ) const [inline]`

Finds an entry by `__order`. Returns a `const_iterator` to the entry with the `__order` order, or a `const_iterator` to the container object's end if order is at least the size of the container object.

Definition at line 72 of file `tree_policy.hpp`.

4.372.2.2 `template<typename Node_Cltr , typename Node_Itr , typename Cmp_Fn , typename _Alloc >  
 tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >::iterator  
 __gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >::find_by_order (`  
`size_type order ) [inline]`

Finds an entry by `__order`. Returns an iterator to the entry with the `__order` order, or an iterator to the container object's end if order is at least the size of the container object.

Definition at line 45 of file `tree_policy.hpp`.

4.372.2.3 `template<typename Node_Cltr , typename Node_Itr , typename Cmp_Fn , typename _Alloc > void  
 __gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >::operator() (`  
`node_iterator node_it, node_const_iterator end_nd_it ) const [inline], [protected]`

Updates the rank of a node through a `node_iterator` `node_it`; `end_nd_it` is the end node iterator.

Definition at line 108 of file `tree_policy.hpp`.

4.372.2.4 `template<typename Node_Cltr , typename Node_Itr , typename Cmp_Fn , typename _Alloc >  
 tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >::size_type  
 __gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >::order_of_key (`  
`key_const_reference r_key ) const [inline]`

Returns the order of a key within a sequence. For example, if `r_key` is the smallest key, this method will return 0; if `r_key` is a key between the smallest and next key, this method will return 1; if `r_key` is a key larger than the largest key, this method will return the size of `r_c`.

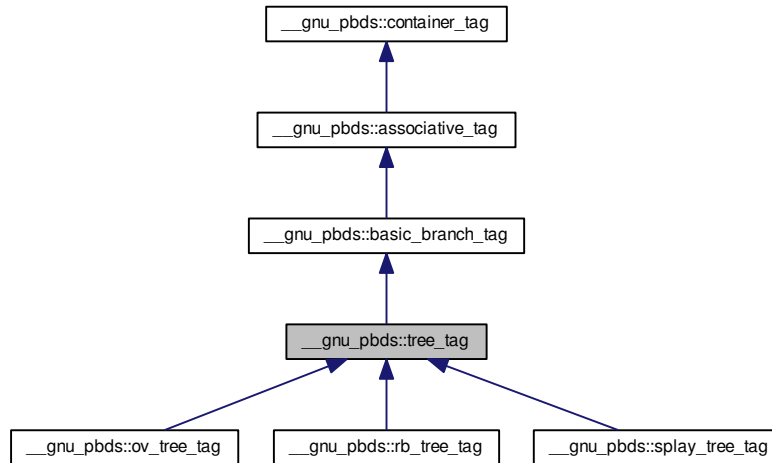
Definition at line 78 of file `tree_policy.hpp`.

The documentation for this class was generated from the following file:

- [tree\\_policy.hpp](#)

## 4.373 \_\_gnu\_pbds::tree\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::tree\_tag:



## 4.373.1 Detailed Description

Basic tree structure.

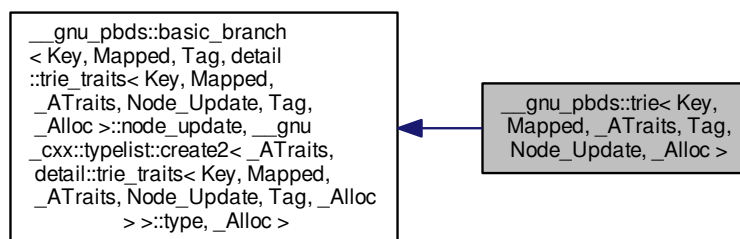
Definition at line 150 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.374 \_\_gnu\_pbds::trie&lt; Key, Mapped, \_ATraits, Tag, Node\_Update, \_Alloc &gt; Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::trie< Key, Mapped, \_ATraits, Tag, Node\_Update, \_Alloc >:



## Public Types

- typedef `_ATraits` [access\\_traits](#)
- typedef [detail::trie\\_traits](#)< Key, Mapped, `_ATraits`, `Node_Update`, `Tag`, `_Alloc` >::`node_update` **node\_update**

## Public Member Functions

- [trie](#) (const [access\\_traits](#) &t)
- template<typename It > [trie](#) (It first, It last)
- template<typename It > [trie](#) (It first, It last, const [access\\_traits](#) &t)
- **trie** (const [trie](#) &other)
- [trie](#) & **operator=** (const [trie](#) &other)
- void **swap** ([trie](#) &other)

## 4.374.1 Detailed Description

```
template<typename Key, typename Mapped, typename _ATraits = typename detail::default_trie_access_traits<Key>::type, typename
Tag = pat_trie_tag, template< typename Node_Cltr, typename Node_Itr, typename _ATraits_, typename _Alloc_ > class Node_Update
= null_node_update, typename _Alloc = std::allocator<char>>> class __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update,
_Alloc >
```

A trie-based container.

## Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>_ATraits</i>	Element access traits.
<i>Tag</i>	Instantiating data structure type, see <code>container_tag</code> .
<i>Node_Update</i>	Updates trie internal-nodes, restores invariants when invalidated. XXX See design↵ ::tree-based-containersnode invariants.
<i>_Alloc</i>	Allocator type.

Base tag choice is `pat_trie_tag`.

Base is `basic_branch`.

Definition at line 731 of file `assoc_container.hpp`.

## 4.374.2 Member Typedef Documentation

4.374.2.1 `template<typename Key , typename Mapped , typename _ATraits = typename detail::default_trie_access_traits<↵  
Key>::type, typename Tag = pat_trie_tag, template< typename Node_Cltr, typename Node_Itr, typename _ATraits_,  
typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>> typedef _ATraits  
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::access_traits`

Element access traits type.

Definition at line 738 of file `assoc_container.hpp`.

## 4.374.3 Constructor &amp; Destructor Documentation

```
4.374.3.1  template<typename Key , typename Mapped , typename _ATraits = typename detail::default_trie_access_traits<Key>↵
::type, typename Tag = pat_trie_tag, template< typename Node_Cltr, typename Node_Itr, typename _ATraits_, typename
_Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>> __gnu_pbds::trie<
Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie ( const access_traits & t )  [inline]
```

Constructor taking some policy objects. `r_access_traits` will be copied by the `_ATraits` object of the container object.

Definition at line 744 of file `assoc_container.hpp`.

```
4.374.3.2  template<typename Key , typename Mapped , typename _ATraits = typename detail::default_trie_access_traits<Key>↵
::type, typename Tag = pat_trie_tag, template< typename Node_Cltr, typename Node_Itr, typename _ATraits_, typename
_Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>> template<typename It >
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie ( It first, It last )  [inline]
```

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

Definition at line 751 of file `assoc_container.hpp`.

```
4.374.3.3  template<typename Key , typename Mapped , typename _ATraits = typename detail::default_trie_access_traits<Key>↵
::type, typename Tag = pat_trie_tag, template< typename Node_Cltr, typename Node_Itr, typename _ATraits_, typename
_Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>> template<typename It >
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie ( It first, It last, const access_traits &
t )  [inline]
```

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

Definition at line 758 of file `assoc_container.hpp`.

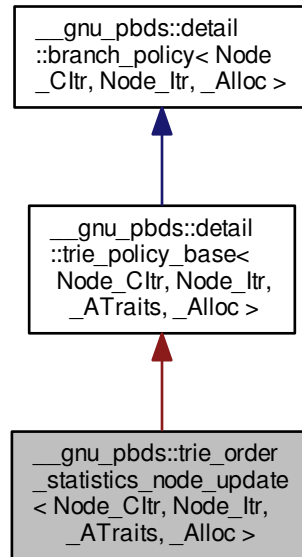
The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)



#### 4.375 `__gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >`:



#### Public Types

- typedef `access_traits::const_iterator` **a\_const\_iterator**
- typedef `_ATraits` **access\_traits**
- typedef `_Alloc` **allocator\_type**
- typedef `node_const_iterator::value_type` **const\_iterator**
- typedef `node_iterator::value_type` **iterator**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_type` **key\_type**
- typedef `size_type` **metadata\_type**
- typedef `Node_Cltr` **node\_const\_iterator**
- typedef `Node_Itr` **node\_iterator**
- typedef `allocator_type::size_type` **size\_type**

#### Public Member Functions

- `const_iterator` [find\\_by\\_order](#) (`size_type`) const
- `iterator` [find\\_by\\_order](#) (`size_type`)
- `size_type` [order\\_of\\_key](#) (`key_const_reference`) const
- `size_type` [order\\_of\\_prefix](#) (`a_const_iterator`, `a_const_iterator`) const

## Protected Member Functions

- void `operator()` (node\_iterator, node\_const\_iterator) const

## Private Types

- typedef Node\_Itr::value\_type **it\_type**
- typedef remove\_const< key\_type >::type **rkkey\_type**
- typedef remove\_const< value\_type >::type **rcvalue\_type**
- typedef \_Alloc::template rebind< rkkey\_type >::other **rebind\_k**
- typedef \_Alloc::template rebind< rcvalue\_type >::other **rebind\_v**
- typedef rebind\_v::reference **reference**
- typedef std::iterator\_traits< it\_type >::value\_type **value\_type**

## Private Member Functions

- virtual const\_iterator **end** () const =0
- it\_type **end\_iterator** () const
- virtual const access\_traits & **get\_access\_traits** () const =0

## Static Private Member Functions

- static size\_type **common\_prefix\_len** (node\_iterator, e\_const\_iterator, e\_const\_iterator, const access\_traits &)
- static key\_const\_reference **extract\_key** (const\_reference r\_val)
- static iterator **leftmost\_it** (node\_iterator)
- static bool **less** (e\_const\_iterator, e\_const\_iterator, e\_const\_iterator, e\_const\_iterator, const access\_traits &)
- static iterator **rightmost\_it** (node\_iterator)

### 4.375.1 Detailed Description

template<typename Node\_Cltr, typename Node\_Itr, typename \_ATraits, typename \_Alloc>class `__gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >`

Functor updating ranks of entrees.

Definition at line 253 of file `trie_policy.hpp`.

### 4.375.2 Member Function Documentation

4.375.2.1 `template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc >`  
`trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::const_iterator`  
`__gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::find_by_order (`  
`size_type order ) const` `[inline]`

Finds an entry by `__order`. Returns a `const_iterator` to the entry with the `__order` order, or a `const_iterator` to the container object's end if order is at least the size of the container object.

Definition at line 79 of file `trie_policy.hpp`.

```
4.375.2.2  template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc
> trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::iterator
__gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::find_by_order (
size_type order )  [inline]
```

Finds an entry by `__order`. Returns an iterator to the entry with the `__order` order, or an iterator to the container object's end if order is at least the size of the container object.

Definition at line 45 of file `trie_policy.hpp`.

```
4.375.2.3  template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc > void
__gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::operator() (
node_iterator nd_it, node_const_iterator ) const  [inline],[protected]
```

Updates the rank of a node through a `node_iterator` `node_it`; `end_nd_it` is the end node iterator.

Definition at line 152 of file `trie_policy.hpp`.

```
4.375.2.4  template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc >
trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::size_type
__gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::order_of_key (
key_const_reference r_key ) const  [inline]
```

Returns the order of a key within a sequence. For example, if `r_key` is the smallest key, this method will return 0; if `r_key` is a key between the smallest and next key, this method will return 1; if `r_key` is a key larger than the largest key, this method will return the size of `r_c`.

Definition at line 85 of file `trie_policy.hpp`.

```
4.375.2.5  template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc >
trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::size_type
__gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::order_of_prefix (
a_const_iterator b, a_const_iterator e ) const  [inline]
```

Returns the order of a prefix within a sequence. For example, if `[b, e]` is the smallest prefix, this method will return 0; if `r_key` is a key between the smallest and next key, this method will return 1; if `r_key` is a key larger than the largest key, this method will return the size of `r_c`.

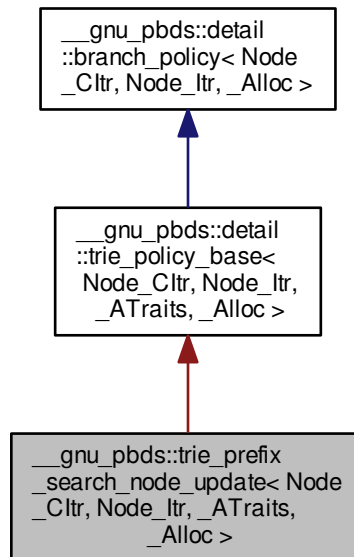
Definition at line 96 of file `trie_policy.hpp`.

The documentation for this class was generated from the following file:

- [trie\\_policy.hpp](#)

## 4.376 `__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >`:



### Public Types

- typedef `access_traits::const_iterator` [a\\_const\\_iterator](#)
- typedef `_ATraits` [access\\_traits](#)
- typedef `_Alloc` [allocator\\_type](#)
- typedef `node_const_iterator::value_type` **const\_iterator**
- typedef `node_iterator::value_type` **iterator**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_type` **key\_type**
- typedef [null\\_type](#) **metadata\_type**
- typedef `Node_Cltr` **node\_const\_iterator**
- typedef `Node_Itr` **node\_iterator**
- typedef `allocator_type::size_type` [size\\_type](#)

### Public Member Functions

- `std::pair< const_iterator, const_iterator >` [prefix\\_range](#) (`key_const_reference`) const
- `std::pair< iterator, iterator >` [prefix\\_range](#) (`key_const_reference`)
- `std::pair< const_iterator, const_iterator >` [prefix\\_range](#) ([a\\_const\\_iterator](#), [a\\_const\\_iterator](#)) const
- `std::pair< iterator, iterator >` [prefix\\_range](#) ([a\\_const\\_iterator](#), [a\\_const\\_iterator](#))

### Protected Member Functions

- void [operator\(\)](#) (node\_iterator node\_it, node\_const\_iterator end\_nd\_it) const

### Private Types

- typedef rebind\_v::const\_pointer **const\_pointer**
- typedef rebind\_v::const\_reference **const\_reference**
- typedef Node\_itr::value\_type **it\_type**
- typedef remove\_const< key\_type >::type **rckey\_type**
- typedef remove\_const< value\_type >::type **rcvalue\_type**
- typedef \_Alloc::template rebind< rckey\_type >::other **rebind\_k**
- typedef \_Alloc::template rebind< rcvalue\_type >::other **rebind\_v**
- typedef rebind\_v::reference **reference**
- typedef std::iterator\_traits< it\_type >::value\_type **value\_type**

### Private Member Functions

- it\_type **end\_iterator** () const

### Static Private Member Functions

- static [size\\_type](#) **common\_prefix\_len** (node\_iterator, e\_const\_iterator, e\_const\_iterator, const [access\\_traits](#) &)
- static key\_const\_reference **extract\_key** (const\_reference r\_val)
- static iterator **leftmost\_it** (node\_iterator)
- static bool **less** (e\_const\_iterator, e\_const\_iterator, e\_const\_iterator, e\_const\_iterator, const [access\\_traits](#) &)
- static iterator **rightmost\_it** (node\_iterator)

#### 4.376.1 Detailed Description

```
template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>class __gnu_pbds::trie_prefix_search_↵
node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >
```

A node updatator that allows tries to be searched for the range of values that match a certain prefix.

Definition at line 155 of file trie\_policy.hpp.

#### 4.376.2 Member Typedef Documentation

4.376.2.1 `template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc> typedef access_traits::const_iterator __gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::a_const_iterator`

Const element iterator.

Definition at line 168 of file trie\_policy.hpp.

4.376.2.2 `template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc> typedef _ATraits __gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::access_traits`

Element access traits.

Definition at line 165 of file trie\_policy.hpp.

4.376.2.3 `template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc> typedef _Alloc  
__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::allocator_type`

`_Alloc` type.

Definition at line 171 of file `trie_policy.hpp`.

4.376.2.4 `template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc> typedef  
allocator_type::size_type __gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc  
>::size_type`

Size type.

Definition at line 174 of file `trie_policy.hpp`.

#### 4.376.3 Member Function Documentation

4.376.3.1 `template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc > void  
__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::operator() (  
node_iterator node_it, node_const_iterator end_nd_it) const [inline], [protected]`

Called to update a node's metadata.

Definition at line 139 of file `trie_policy.hpp`.

4.376.3.2 `template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc > std::pair<  
typename trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::const_iterator,  
typename trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::const_iterator >  
__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::prefix_range (  
key_const_reference r_key) const`

Finds the const iterator range corresponding to all values whose prefixes match `r_key`.

Definition at line 47 of file `trie_policy.hpp`.

4.376.3.3 `template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc > std::pair<  
typename trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::iterator,  
typename trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::iterator >  
__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::prefix_range (  
key_const_reference r_key)`

Finds the iterator range corresponding to all values whose prefixes match `r_key`.

Definition at line 58 of file `trie_policy.hpp`.

4.376.3.4 `template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc > std::pair<  
typename trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::const_iterator,  
typename trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::const_iterator >  
__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::prefix_range (  
a_const_iterator b, a_const_iterator e) const`

Finds the const iterator range corresponding to all values whose prefixes match `[b, e)`.

Definition at line 69 of file `trie_policy.hpp`.

```
4.376.3.5  template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc > std::pair<
            typename trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::iterator,
            typename trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::iterator >
            __gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::prefix_range (
            a_const_iterator b, a_const_iterator e )
```

Finds the iterator range corresponding to all values whose prefixes match [b, e).

Definition at line 84 of file trie\_policy.hpp.

The documentation for this class was generated from the following file:

- [trie\\_policy.hpp](#)

#### 4.377 \_\_gnu\_pbds::trie\_string\_access\_traits< String, Min\_E\_Val, Max\_E\_Val, Reverse, \_Alloc > Struct Template Reference

##### Public Types

- enum { **reverse** }
- enum { **min\_e\_val**, **max\_e\_val**, **max\_size** }
- typedef \_Alloc::template rebind< key\_type > **\_\_rebind\_k**
- typedef detail::\_\_conditional\_type< Reverse, typename String::const\_reverse\_iterator, typename String::const\_iterator >::\_\_type **const\_iterator**
- typedef std::iterator\_traits< **const\_iterator** >::value\_type **e\_type**
- typedef \_\_rebind\_k::other::const\_reference **key\_const\_reference**
- typedef String **key\_type**
- typedef \_Alloc::size\_type **size\_type**

##### Static Public Member Functions

- static **const\_iterator begin** (key\_const\_reference)
- static size\_type **e\_pos** (**e\_type** e)
- static **const\_iterator end** (key\_const\_reference)

##### 4.377.1 Detailed Description

```
template<typename String = std::string, typename String::value_type Min_E_Val = detail::__numeric_traits<typename String::value_type>::__min,
typename String::value_type Max_E_Val = detail::__numeric_traits<typename String::value_type>::__max, bool Reverse = false,
typename _Alloc = std::allocator<char>> struct __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >
```

Element access traits for string types.

##### Template Parameters

<i>String</i>	String type.
<i>Min_E_Val</i>	Minimal element value.

<i>Max_E_Val</i>	Maximum element value.
<i>Reverse</i>	Reverse iteration should be used. Default: false.
<i>_Alloc</i>	Allocator type.

Definition at line 74 of file `trie_policy.hpp`.

#### 4.377.2 Member Typedef Documentation

4.377.2.1 `template<typename String = std::string, typename String::value_type Min_E_Val = detail::__numeric_traits<typename String::value_type>::__min, typename String::value_type Max_E_Val = detail::__numeric_traits<typename String::value_type>::__max, bool Reverse = false, typename _Alloc = std::allocator<char>> typedef detail::__conditional_type<Reverse, typename String::const_reverse_iterator, typename String::const_iterator>::__type __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::const_iterator`

Element const iterator type.

Definition at line 90 of file `trie_policy.hpp`.

4.377.2.2 `template<typename String = std::string, typename String::value_type Min_E_Val = detail::__numeric_traits<typename String::value_type>::__min, typename String::value_type Max_E_Val = detail::__numeric_traits<typename String::value_type>::__max, bool Reverse = false, typename _Alloc = std::allocator<char>> typedef std::iterator_traits<const_iterator>::value_type __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::e_type`

Element type.

Definition at line 93 of file `trie_policy.hpp`.

#### 4.377.3 Member Function Documentation

4.377.3.1 `template<typename String , typename String::value_type Min_E_Val, typename String::value_type Max_E_Val, bool Reverse, typename _Alloc > trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::const_iterator __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::begin ( key_const_reference r_key ) [inline],[static]`

Returns a `const_iterator` to the first element of `key_const_reference` agumnet.

Definition at line 57 of file `trie_policy.hpp`.

4.377.3.2 `template<typename String , typename String::value_type Min_E_Val, typename String::value_type Max_E_Val, bool Reverse, typename _Alloc > trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::size_type __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::e_pos ( e_type e ) [inline],[static]`

Maps an element to a position.

Definition at line 49 of file `trie_policy.hpp`.

4.377.3.3 `template<typename String , typename String::value_type Min_E_Val, typename String::value_type Max_E_Val, bool Reverse, typename _Alloc > trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::const_iterator __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::end ( key_const_reference r_key ) [inline],[static]`

Returns a `const_iterator` to the after-last element of `key_const_reference` argument.

Definition at line 65 of file `trie_policy.hpp`.

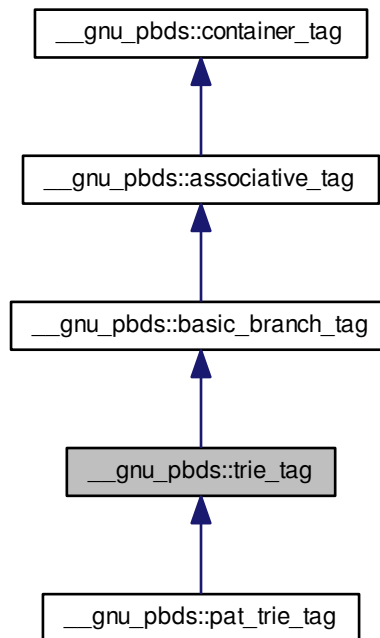


The documentation for this struct was generated from the following file:

- [trie\\_policy.hpp](#)

#### 4.378 \_\_gnu\_pbds::trie\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::trie\_tag:



##### 4.378.1 Detailed Description

Basic trie structure.

Definition at line 162 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.379 \_\_gnu\_pbds::trivial\_iterator\_tag Struct Reference

##### 4.379.1 Detailed Description

A trivial iterator tag. Signifies that the iterators has none of `std::iterators`'s movement abilities.

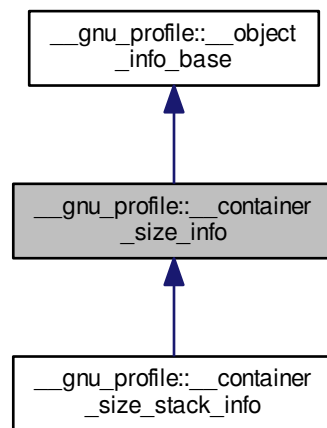
Definition at line 75 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.380 `__gnu_profile::__container_size_info` Class Reference

Inheritance diagram for `__gnu_profile::__container_size_info`:



#### Public Member Functions

- `__container_size_info` (const [\\_\\_container\\_size\\_info](#) &\_\_o)
- `__container_size_info` (`__stack_t` \_\_stack, `std::size_t` \_\_num)
- `std::string` `__advice` () const
- void `__destruct` (`std::size_t` \_\_num, `std::size_t` \_\_inum)
- bool `__is_valid` () const
- float `__magnitude` () const
- void `__merge` (const [\\_\\_container\\_size\\_info](#) &\_\_o)
- void `__resize` (`std::size_t` \_\_from, `std::size_t` \_\_to)
- float `__resize_cost` (`std::size_t` \_\_from, `std::size_t` \_\_to)
- `__stack_t` `__stack` () const
- void `__write` (FILE \* \_\_f) const

#### Protected Attributes

- `__stack_t` `M_stack`
- bool `M_valid`

#### 4.380.1 Detailed Description

A container size instrumentation line in the object table.

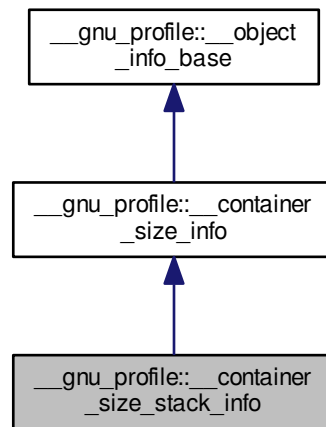
Definition at line 42 of file profiler\_container\_size.h.

The documentation for this class was generated from the following file:

- [profiler\\_container\\_size.h](#)

#### 4.381 \_\_gnu\_profile::\_\_container\_size\_stack\_info Class Reference

Inheritance diagram for \_\_gnu\_profile::\_\_container\_size\_stack\_info:



#### Public Member Functions

- **\_\_container\_size\_stack\_info** (const [\\_\\_container\\_size\\_info](#) &\_\_o)
- **std::string \_\_advice** () const
- void **\_\_destruct** (std::size\_t \_\_num, std::size\_t \_\_inum)
- bool **\_\_is\_valid** () const
- float **\_\_magnitude** () const
- void **\_\_merge** (const [\\_\\_container\\_size\\_info](#) &\_\_o)
- void **\_\_resize** (std::size\_t \_\_from, std::size\_t \_\_to)
- float **\_\_resize\_cost** (std::size\_t \_\_from, std::size\_t)
- **\_\_stack\_t \_\_stack** () const
- void **\_\_write** (FILE \*\_\_f) const

#### Protected Attributes

- **\_\_stack\_t \_M\_stack**
- bool **\_M\_valid**

## 4.381.1 Detailed Description

A container size instrumentation line in the stack table.

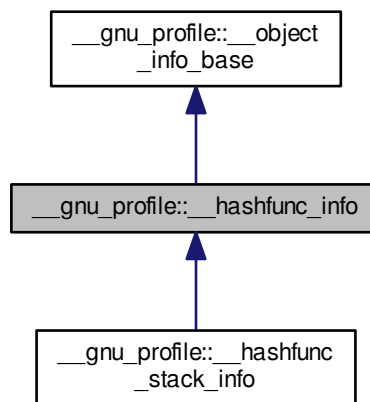
Definition at line 154 of file profiler\_container\_size.h.

The documentation for this class was generated from the following file:

- [profiler\\_container\\_size.h](#)

## 4.382 \_\_gnu\_profile::\_\_hashfunc\_info Class Reference

Inheritance diagram for \_\_gnu\_profile::\_\_hashfunc\_info:



## Public Member Functions

- `__hashfunc_info` (const [\\_\\_hashfunc\\_info](#) &\_\_o)
- `__hashfunc_info` (\_\_stack\_t \_\_stack)
- `std::string __advice` () const
- void `__destruct` (std::size\_t \_\_chain, std::size\_t \_\_accesses, std::size\_t \_\_hops)
- bool `__is_valid` () const
- float `__magnitude` () const
- void `__merge` (const [\\_\\_hashfunc\\_info](#) &\_\_o)
- `__stack_t __stack` () const
- void `__write` (FILE \*\_\_f) const

## Protected Attributes

- `__stack_t M_stack`
- bool `M_valid`

#### 4.382.1 Detailed Description

A hash performance instrumentation line in the object table.

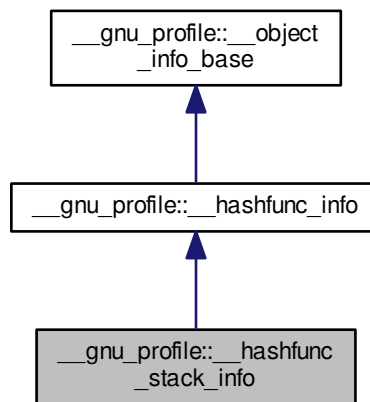
Definition at line 40 of file profiler\_hash\_func.h.

The documentation for this class was generated from the following file:

- [profiler\\_hash\\_func.h](#)

#### 4.383 \_\_gnu\_profile::\_\_hashfunc\_stack\_info Class Reference

Inheritance diagram for \_\_gnu\_profile::\_\_hashfunc\_stack\_info:



#### Public Member Functions

- **\_\_hashfunc\_stack\_info** (const [\\_\\_hashfunc\\_info](#) &\_\_o)
- [std::string](#) **\_\_advice** () const
- void **\_\_destruct** (std::size\_t \_\_chain, std::size\_t \_\_accesses, std::size\_t \_\_hops)
- bool **\_\_is\_valid** () const
- float **\_\_magnitude** () const
- void **\_\_merge** (const [\\_\\_hashfunc\\_info](#) &\_\_o)
- [\\_\\_stack\\_t](#) **\_\_stack** () const
- void **\_\_write** (FILE \*\_\_f) const

#### Protected Attributes

- [\\_\\_stack\\_t](#) **\_M\_stack**
- bool **\_M\_valid**

## 4.383.1 Detailed Description

A hash performance instrumentation line in the stack table.

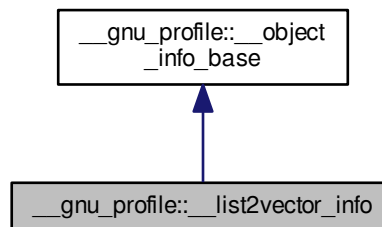
Definition at line 95 of file `profiler_hash_func.h`.

The documentation for this class was generated from the following file:

- [profiler\\_hash\\_func.h](#)

4.384 `__gnu_profile::__list2vector_info` Class Reference

Inheritance diagram for `__gnu_profile::__list2vector_info`:



## Public Member Functions

- `__list2vector_info` (`__stack_t __stack`)
- `__list2vector_info` (`const __list2vector_info &__o`)
- `std::string __advice` () const
- `bool __is_valid` ()
- `bool __is_valid` () const
- `std::size_t __iterate` ()
- `float __list_cost` ()
- `float __magnitude` () const
- `void __merge` (`const __list2vector_info &__o`)
- `void __opr_insert` (`std::size_t __shift`, `std::size_t __size`)
- `void __opr_iterate` (`std::size_t __num`)
- `std::size_t __resize` ()
- `void __resize` (`std::size_t __from`, `std::size_t`)
- `void __set_invalid` ()
- `void __set_list_cost` (`float __lc`)
- `void __set_vector_cost` (`float __vc`)
- `std::size_t __shift_count` ()
- `__stack_t __stack` () const
- `void __write` (`FILE *__f`) const

## Protected Attributes

- `__stack_t __M_stack`

## 4.384.1 Detailed Description

A list-to-vector instrumentation line in the object table.

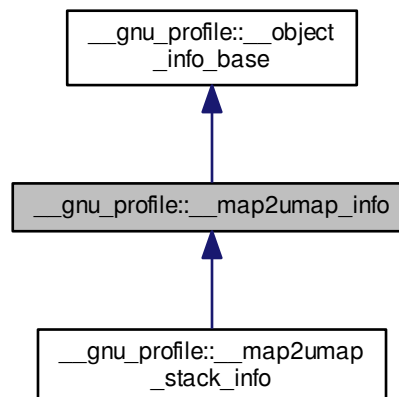
Definition at line 42 of file `profiler_list_to_vector.h`.

The documentation for this class was generated from the following file:

- [profiler\\_list\\_to\\_vector.h](#)

4.385 `__gnu_profile::__map2umap_info` Class Reference

Inheritance diagram for `__gnu_profile::__map2umap_info`:



## Public Member Functions

- `__map2umap_info (__stack_t __stack)`
- `__map2umap_info (const __map2umap_info &__o)`
- `std::string __advice () const`
- `bool __is_valid () const`
- `float __magnitude () const`
- `void __merge (const __map2umap_info &__o)`
- `void __record_erase (std::size_t __size, std::size_t __count)`
- `void __record_find (std::size_t __size)`
- `void __record_insert (std::size_t __size, std::size_t __count)`
- `void __record_invalidate ()`
- `void __record_iterate (std::size_t __count)`

- `__stack_t __stack () const`
- `void __write (FILE *__f) const`

#### Protected Attributes

- `__stack_t _M_stack`

#### 4.385.1 Detailed Description

A map-to-unordered\_map instrumentation line in the object table.

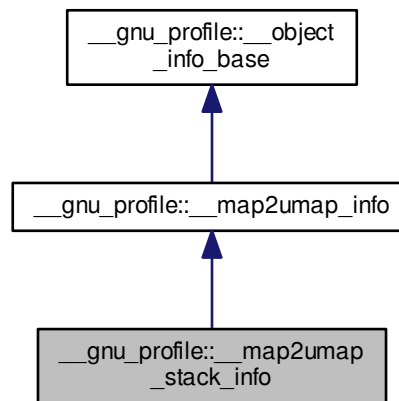
Definition at line 66 of file `profiler_map_to_unordered_map.h`.

The documentation for this class was generated from the following file:

- [profiler\\_map\\_to\\_unordered\\_map.h](#)

#### 4.386 \_\_gnu\_profile::\_\_map2umap\_stack\_info Class Reference

Inheritance diagram for `__gnu_profile::__map2umap_stack_info`:



#### Public Member Functions

- `__map2umap_stack_info (const __map2umap_info &__o)`
- `std::string __advice () const`
- `bool __is_valid () const`
- `float __magnitude () const`
- `void __merge (const __map2umap_info &__o)`
- `void __record_erase (std::size_t __size, std::size_t __count)`
- `void __record_find (std::size_t __size)`



- void **\_\_record\_insert** (std::size\_t \_\_size, std::size\_t \_\_count)
- void **\_\_record\_invalidate** ()
- void **\_\_record\_iterate** (std::size\_t \_\_count)
- \_\_stack\_t **\_\_stack** () const
- void **\_\_write** (FILE \*\_\_f) const

#### Protected Attributes

- \_\_stack\_t **\_M\_stack**

#### 4.386.1 Detailed Description

A map-to-unordered\_map instrumentation line in the stack table.

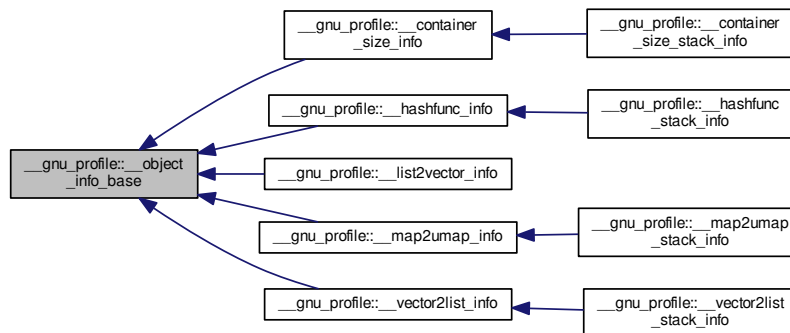
Definition at line 170 of file profiler\_map\_to\_unordered\_map.h.

The documentation for this class was generated from the following file:

- [profiler\\_map\\_to\\_unordered\\_map.h](#)

#### 4.387 \_\_gnu\_profile::\_\_object\_info\_base Class Reference

Inheritance diagram for \_\_gnu\_profile::\_\_object\_info\_base:



#### Public Member Functions

- **\_\_object\_info\_base** (\_\_stack\_t \_\_stack)
- **\_\_object\_info\_base** (const [\\_\\_object\\_info\\_base](#) &\_\_o)
- bool **\_\_is\_valid** () const
- \_\_stack\_t **\_\_stack** () const
- virtual void **\_\_write** (FILE \*\_\_f) const =0

#### Protected Attributes

- \_\_stack\_t **\_M\_stack**
- bool **\_M\_valid**

## 4.387.1 Detailed Description

Base class for a line in the object table.

Definition at line 123 of file `profiler_node.h`.

The documentation for this class was generated from the following file:

- [profiler\\_node.h](#)

4.388 `__gnu_profile::__reentrance_guard` Struct Reference

## Static Public Member Functions

- static bool `__get_in` ()
- static bool & `__inside` ()

## 4.388.1 Detailed Description

Reentrance guard.

Mechanism to protect all `__gnu_profile` operations against recursion, multithreaded and exception reentrance.

Definition at line 58 of file `profiler.h`.

The documentation for this struct was generated from the following file:

- [profiler.h](#)

4.389 `__gnu_profile::__stack_hash` Class Reference

## Public Member Functions

- `std::size_t operator()` (`__stack_t __s`) const
- bool `operator()` (`__stack_t __stack1`, `__stack_t __stack2`) const

## 4.389.1 Detailed Description

Hash function for summary trace using call stack as index.

Definition at line 89 of file `profiler_node.h`.

The documentation for this class was generated from the following file:

- [profiler\\_node.h](#)

4.390 `__gnu_profile::__stack_info_base<__object_info>` Class Template Reference

## Public Member Functions

- `__stack_info_base` (const `__object_info` & `__info`)=0
- virtual const char \* `__get_id` () const =0
- virtual float `__magnitude` () const =0
- void `__merge` (const `__object_info` & `__info`)=0

#### 4.390.1 Detailed Description

```
template<typename __object_info>class __gnu_profile::__stack_info_base< __object_info >
```

Base class for a line in the stack table.

Definition at line 154 of file profiler\_node.h.

The documentation for this class was generated from the following file:

- [profiler\\_node.h](#)

#### 4.391 \_\_gnu\_profile::\_\_trace\_base< \_\_object\_info, \_\_stack\_info > Class Template Reference

##### Public Member Functions

- void **\_\_add\_object** (\_\_object\_t object, \_\_object\_info \_\_info)
- void **\_\_collect\_warnings** (\_\_warning\_vector\_t &\_\_warnings)
- \_\_object\_info \* **\_\_get\_object\_info** (\_\_object\_t \_\_object)
- void **\_\_retire\_object** (\_\_object\_t \_\_object)
- void **\_\_write** (FILE \*\_\_f)

##### Protected Attributes

- const char \* **\_\_id**

#### 4.391.1 Detailed Description

```
template<typename __object_info, typename __stack_info>class __gnu_profile::__trace_base< __object_info, __stack_info >
```

Base class for all trace producers.

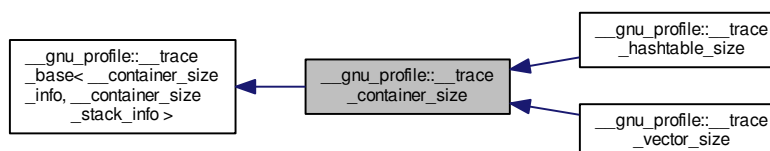
Definition at line 183 of file profiler\_trace.h.

The documentation for this class was generated from the following file:

- [profiler\\_trace.h](#)

#### 4.392 \_\_gnu\_profile::\_\_trace\_container\_size Class Reference

Inheritance diagram for \_\_gnu\_profile::\_\_trace\_container\_size:



## Public Member Functions

- void `__add_object` (`__object_t` object, `__container_size_info__info`)
- void `__collect_warnings` (`__warning_vector_t` &`__warnings`)
- void `__construct` (const void \*`__obj`, `std::size_t` `__inum`)
- void `__destruct` (const void \*`__obj`, `std::size_t` `__num`, `std::size_t` `__inum`)
- `__container_size_info` \* `__get_object_info` (`__object_t` `__object`)
- void `__insert` (const `__object_t` `__obj`, `__stack_t` `__stack`, `std::size_t` `__num`)
- void `__resize` (const void \*`__obj`, int `__from`, int `__to`)
- void `__retire_object` (`__object_t` `__object`)
- void `__write` (FILE \*`__f`)

## Protected Attributes

- const char \* `__id`

## 4.392.1 Detailed Description

Container size instrumentation trace producer.

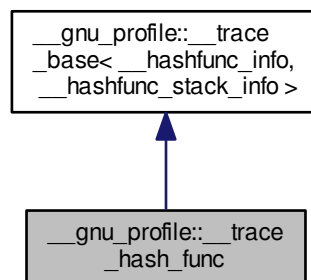
Definition at line 164 of file `profiler_container_size.h`.

The documentation for this class was generated from the following file:

- [profiler\\_container\\_size.h](#)

4.393 `__gnu_profile::__trace_hash_func` Class Reference

Inheritance diagram for `__gnu_profile::__trace_hash_func`:



## Public Member Functions

- void `__add_object` (`__object_t` object, `__hashfunc_info__info`)
- void `__collect_warnings` (`__warning_vector_t` &`__warnings`)

- void **\_\_destruct** (const void \* \_\_obj, std::size\_t \_\_chain, std::size\_t \_\_accesses, std::size\_t \_\_hops)
- [\\_\\_hashfunc\\_info](#) \* **\_\_get\_object\_info** (\_\_object\_t \_\_object)
- void **\_\_insert** (\_\_object\_t \_\_obj, \_\_stack\_t \_\_stack)
- void **\_\_retire\_object** (\_\_object\_t \_\_object)
- void **\_\_write** (FILE \* \_\_f)

#### Protected Attributes

- const char \* **\_\_id**

#### 4.393.1 Detailed Description

Hash performance instrumentation producer.

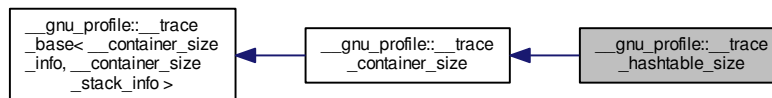
Definition at line 105 of file profiler\_hash\_func.h.

The documentation for this class was generated from the following file:

- [profiler\\_hash\\_func.h](#)

#### 4.394 `__gnu_profile::__trace_hashtable_size` Class Reference

Inheritance diagram for `__gnu_profile::__trace_hashtable_size`:



#### Public Member Functions

- void **\_\_add\_object** (\_\_object\_t object, \_\_container\_size\_info \_\_info)
- void **\_\_collect\_warnings** (\_\_warning\_vector\_t & \_\_warnings)
- void **\_\_construct** (const void \* \_\_obj, std::size\_t \_\_inum)
- void **\_\_destruct** (const void \* \_\_obj, std::size\_t \_\_num, std::size\_t \_\_inum)
- [\\_\\_container\\_size\\_info](#) \* **\_\_get\_object\_info** (\_\_object\_t \_\_object)
- void **\_\_insert** (const \_\_object\_t \_\_obj, \_\_stack\_t \_\_stack, std::size\_t \_\_num)
- void **\_\_resize** (const void \* \_\_obj, int \_\_from, int \_\_to)
- void **\_\_retire\_object** (\_\_object\_t \_\_object)
- void **\_\_write** (FILE \* \_\_f)

#### Protected Attributes

- const char \* **\_\_id**

## 4.394.1 Detailed Description

Hashtable size instrumentation trace producer.

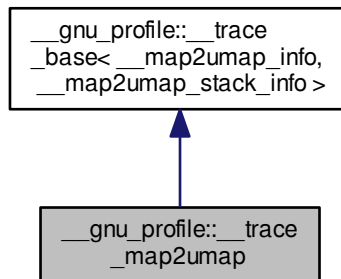
Definition at line 42 of file `profiler_hashtable_size.h`.

The documentation for this class was generated from the following file:

- [profiler\\_hashtable\\_size.h](#)

4.395 `__gnu_profile::__trace_map2umap` Class Reference

Inheritance diagram for `__gnu_profile::__trace_map2umap`:



## Public Member Functions

- void **\_\_add\_object** (`__object_t` object, `__map2umap_info` info)
- void **\_\_collect\_warnings** (`__warning_vector_t` &\_\_warnings)
- `__map2umap_info` \* **\_\_get\_object\_info** (`__object_t` \_\_object)
- void **\_\_retire\_object** (`__object_t` \_\_object)
- void **\_\_write** (`FILE` \* \_\_f)

## Protected Attributes

- const char \* **\_\_id**

## 4.395.1 Detailed Description

Map-to-unordered\_map instrumentation producer.

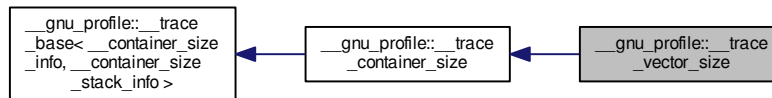
Definition at line 179 of file `profiler_map_to_unordered_map.h`.

The documentation for this class was generated from the following file:

- [profiler\\_map\\_to\\_unordered\\_map.h](#)

### 4.396 `__gnu_profile::__trace_vector_size` Class Reference

Inheritance diagram for `__gnu_profile::__trace_vector_size`:



#### Public Member Functions

- void **\_\_add\_object** (`__object_t` object, `__container_size_info__info`)
- void **\_\_collect\_warnings** (`__warning_vector_t` &`__warnings`)
- void **\_\_construct** (const void \*`__obj`, `std::size_t` `__inum`)
- void **\_\_destruct** (const void \*`__obj`, `std::size_t` `__num`, `std::size_t` `__inum`)
- `__container_size_info` \* **\_\_get\_object\_info** (`__object_t` `__object`)
- void **\_\_insert** (const `__object_t` `__obj`, `__stack_t` `__stack`, `std::size_t` `__num`)
- void **\_\_resize** (const void \*`__obj`, int `__from`, int `__to`)
- void **\_\_retire\_object** (`__object_t` `__object`)
- void **\_\_write** (FILE \*`__f`)

#### Protected Attributes

- const char \* **\_\_id**

#### 4.396.1 Detailed Description

Hashtable size instrumentation trace producer.

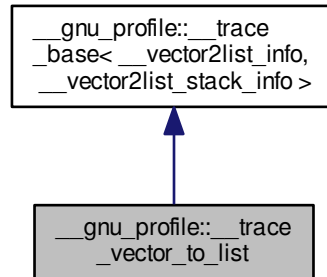
Definition at line 42 of file `profiler_vector_size.h`.

The documentation for this class was generated from the following file:

- [profiler\\_vector\\_size.h](#)

## 4.397 \_\_gnu\_profile::\_\_trace\_vector\_to\_list Class Reference

Inheritance diagram for \_\_gnu\_profile::\_\_trace\_vector\_to\_list:



## Public Member Functions

- void **\_\_add\_object** (\_\_object\_t object, \_\_vector2list\_info \_\_info)
- void **\_\_collect\_warnings** (\_\_warning\_vector\_t & \_\_warnings)
- void **\_\_destruct** (const void \* \_\_obj)
- [\\_\\_vector2list\\_info](#) \* **\_\_find** (const void \* \_\_obj)
- [\\_\\_vector2list\\_info](#) \* **\_\_get\_object\_info** (\_\_object\_t \_\_object)
- void **\_\_insert** (\_\_object\_t \_\_obj, \_\_stack\_t \_\_stack)
- void **\_\_invalid\_operator** (const void \* \_\_obj)
- float **\_\_list\_cost** (std::size\_t \_\_shift, std::size\_t \_\_iterate, std::size\_t \_\_resize)
- void **\_\_opr\_find** (const void \* \_\_obj, std::size\_t \_\_size)
- void **\_\_opr\_insert** (const void \* \_\_obj, std::size\_t \_\_pos, std::size\_t \_\_num)
- void **\_\_opr\_iterate** (const void \* \_\_obj, std::size\_t \_\_num)
- void **\_\_resize** (const void \* \_\_obj, std::size\_t \_\_from, std::size\_t \_\_to)
- void **\_\_retire\_object** (\_\_object\_t \_\_object)
- float **\_\_vector\_cost** (std::size\_t \_\_shift, std::size\_t \_\_iterate, std::size\_t \_\_resize)
- void **\_\_write** (FILE \* \_\_f)

## Protected Attributes

- const char \* **\_\_id**

## 4.397.1 Detailed Description

Vector-to-list instrumentation producer.

Definition at line 158 of file `profiler_vector_to_list.h`.

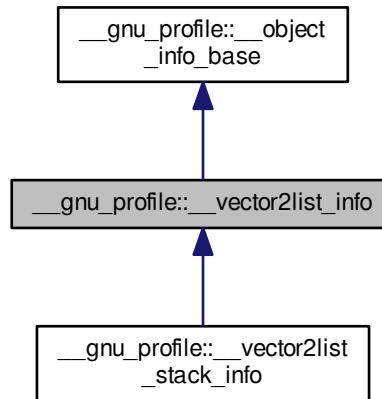
The documentation for this class was generated from the following file:

- [profiler\\_vector\\_to\\_list.h](#)



### 4.398 `__gnu_profile::__vector2list_info` Class Reference

Inheritance diagram for `__gnu_profile::__vector2list_info`:



#### Public Member Functions

- `__vector2list_info` (`__stack_t __stack`)
- `__vector2list_info` (const `__vector2list_info` &`__o`)
- `std::string __advice` () const
- `bool __is_valid` ()
- `bool __is_valid` () const
- `std::size_t __iterate` ()
- `float __list_cost` ()
- `float __magnitude` () const
- `void __merge` (const `__vector2list_info` &`__o`)
- `void __opr_find` (`std::size_t __size`)
- `void __opr_insert` (`std::size_t __pos`, `std::size_t __num`)
- `void __opr_iterate` (`std::size_t __num`)
- `std::size_t __resize` ()
- `void __resize` (`std::size_t __from`, `std::size_t`)
- `void __set_invalid` ()
- `void __set_list_cost` (`float __lc`)
- `void __set_vector_cost` (`float __vc`)
- `std::size_t __shift_count` ()
- `__stack_t __stack` () const
- `void __write` (`FILE * __f`) const

#### Protected Attributes

- `__stack_t __M_stack`

## 4.398.1 Detailed Description

A vector-to-list instrumentation line in the object table.

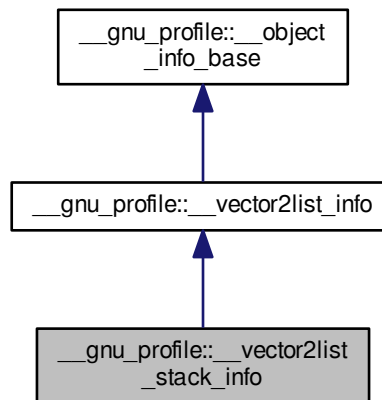
Definition at line 40 of file `profiler_vector_to_list.h`.

The documentation for this class was generated from the following file:

- [profiler\\_vector\\_to\\_list.h](#)

4.399 `__gnu_profile::__vector2list_stack_info` Class Reference

Inheritance diagram for `__gnu_profile::__vector2list_stack_info`:



## Public Member Functions

- `__vector2list_stack_info` (const [\\_\\_vector2list\\_info](#) &\_\_o)
- `std::string __advice` () const
- `bool __is_valid` ()
- `bool __is_valid` () const
- `std::size_t __iterate` ()
- `float __list_cost` ()
- `float __magnitude` () const
- `void __merge` (const [\\_\\_vector2list\\_info](#) &\_\_o)
- `void __opr_find` (std::size\_t \_\_size)
- `void __opr_insert` (std::size\_t \_\_pos, std::size\_t \_\_num)
- `void __opr_iterate` (std::size\_t \_\_num)
- `std::size_t __resize` ()
- `void __resize` (std::size\_t \_\_from, std::size\_t \_\_t)
- `void __set_invalid` ()
- `void __set_list_cost` (float \_\_lc)

- void **\_\_set\_vector\_cost** (float \_\_vc)
- std::size\_t **\_\_shift\_count** ()
- \_\_stack\_t **\_\_stack** () const
- void **\_\_write** (FILE \*\_\_f) const

#### Protected Attributes

- \_\_stack\_t **\_M\_stack**

#### 4.399.1 Detailed Description

A vector-to-list instrumentation line in the stack table.

Definition at line 148 of file profiler\_vector\_to\_list.h.

The documentation for this class was generated from the following file:

- [profiler\\_vector\\_to\\_list.h](#)

#### 4.400 \_\_gnu\_profile::\_\_warning\_data Struct Reference

##### Public Member Functions

- **\_\_warning\_data** (float \_\_m, \_\_stack\_t \_\_c, const char \*\_\_id, const [std::string](#) &\_\_msg)
- bool **operator<** (const [\\_\\_warning\\_data](#) &\_\_other) const

##### Public Attributes

- \_\_stack\_t **\_\_context**
- float **\_\_magnitude**
- const char \* **\_\_warning\_id**
- [std::string](#) **\_\_warning\_message**

#### 4.400.1 Detailed Description

Representation of a warning.

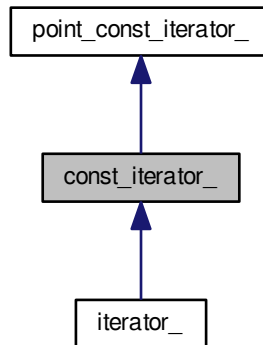
Definition at line 73 of file profiler\_trace.h.

The documentation for this struct was generated from the following file:

- [profiler\\_trace.h](#)

4.401 `const_iterator_` Class Reference

Inheritance diagram for `const_iterator_`:



## Public Types

- `typedef const_pointer_ const\_pointer`
- `typedef const_reference_ const\_reference`
- `typedef _Alloc::difference_type difference\_type`
- `typedef std::forward\_iterator\_tag iterator\_category`
- `typedef pointer_ pointer`
- `typedef reference_ reference`
- `typedef value_type_ value\_type`

## Public Member Functions

- `const\_iterator\_ ()`
- `bool operator!= (const point\_iterator\_ &other) const`
- `bool operator!= (const point\_const\_iterator\_ &other) const`
- `const\_reference operator\* () const`
- `const\_iterator\_ & operator++ ()`
- `const\_iterator\_ operator++ (int)`
- `const\_pointer operator-> () const`
- `bool operator== (const point\_iterator\_ &other) const`
- `bool operator== (const point\_const\_iterator\_ &other) const`

## Protected Types

- `typedef point\_const\_iterator\_ base\_type`

### Protected Member Functions

- [const\\_iterator\\_](#) (const\_pointer\_ p\_value, PB\_DS\_GEN\_POS pos, const PB\_DS\_CLASS\_C\_DEC \*p\_tbl)

### Protected Attributes

- const PB\_DS\_CLASS\_C\_DEC \* [m\\_p\\_tbl](#)
- [const\\_pointer](#) [m\\_p\\_value](#)
- PB\_DS\_GEN\_POS [m\\_pos](#)

### Friends

- class **PB\_DS\_CLASS\_C\_DEC**

#### 4.401.1 Detailed Description

Const range-type iterator.

Definition at line 43 of file unordered\_iterator/const\_iterator.hpp.

#### 4.401.2 Member Typedef Documentation

##### 4.401.2.1 `typedef const_pointer_ const_iterator_::const_pointer`

Iterator's const pointer type.

Definition at line 60 of file unordered\_iterator/const\_iterator.hpp.

##### 4.401.2.2 `typedef const_reference_ const_iterator_::const_reference`

Iterator's const reference type.

Definition at line 66 of file unordered\_iterator/const\_iterator.hpp.

##### 4.401.2.3 `typedef _Alloc::difference_type const_iterator_::difference_type`

Difference type.

Definition at line 51 of file unordered\_iterator/const\_iterator.hpp.

##### 4.401.2.4 `typedef std::forward_iterator_tag const_iterator_::iterator_category`

Category.

Definition at line 48 of file unordered\_iterator/const\_iterator.hpp.

##### 4.401.2.5 `typedef pointer_ const_iterator_::pointer`

Iterator's pointer type.

Definition at line 57 of file unordered\_iterator/const\_iterator.hpp.

##### 4.401.2.6 `typedef reference_ const_iterator_::reference`

Iterator's reference type.

Definition at line 63 of file unordered\_iterator/const\_iterator.hpp.

#### 4.401.2.7 typedef value\_type\_ const\_iterator\_::value\_type

Iterator's value type.

Definition at line 54 of file unordered\_iterator/const\_iterator.hpp.

#### 4.401.3 Constructor & Destructor Documentation

##### 4.401.3.1 const\_iterator\_::const\_iterator\_ ( ) [inline]

Default constructor.

Definition at line 69 of file unordered\_iterator/const\_iterator.hpp.

##### 4.401.3.2 const\_iterator\_::const\_iterator\_ ( const\_pointer\_ p\_value, PB\_DS\_GEN\_POS pos, const PB\_DS\_CLASS\_C\_DEC \* p\_tbl ) [inline], [protected]

Constructor used by the table to initiate the generalized pointer and position (e.g., this is called from within a find() of a table.

Definition at line 97 of file unordered\_iterator/const\_iterator.hpp.

#### 4.401.4 Member Function Documentation

##### 4.401.4.1 bool point\_const\_iterator\_::operator!=( const point\_iterator\_ & other ) const [inline], [inherited]

Compares content (negatively) to a different iterator object.

Definition at line 118 of file unordered\_iterator/point\_const\_iterator.hpp.

##### 4.401.4.2 bool point\_const\_iterator\_::operator!=( const point\_const\_iterator\_ & other ) const [inline], [inherited]

Compares content (negatively) to a different iterator object.

Definition at line 123 of file unordered\_iterator/point\_const\_iterator.hpp.

##### 4.401.4.3 const\_reference point\_const\_iterator\_::operator\*( ) const [inline], [inherited]

Access.

Definition at line 100 of file unordered\_iterator/point\_const\_iterator.hpp.

##### 4.401.4.4 const\_iterator\_ & const\_iterator\_::operator++ ( ) [inline]

Increments.

Definition at line 74 of file unordered\_iterator/const\_iterator.hpp.

References m\_p\_tbl.

##### 4.401.4.5 const\_iterator\_ const\_iterator\_::operator++ ( int ) [inline]

Increments.

Definition at line 82 of file unordered\_iterator/const\_iterator.hpp.

References m\_p\_tbl.

4.401.4.6 `const_pointer point_const_iterator::operator-> ( ) const` `[inline],[inherited]`

Access.

Definition at line 92 of file `unordered_iterator/point_const_iterator.hpp`.

4.401.4.7 `bool point_const_iterator::operator==( const point_iterator_ & other ) const` `[inline],[inherited]`

Compares content to a different iterator object.

Definition at line 108 of file `unordered_iterator/point_const_iterator.hpp`.

4.401.4.8 `bool point_const_iterator::operator==( const point_const_iterator_ & other ) const` `[inline],[inherited]`

Compares content to a different iterator object.

Definition at line 113 of file `unordered_iterator/point_const_iterator.hpp`.

#### 4.401.5 Member Data Documentation

4.401.5.1 `const PB_DS_CLASS_C_DEC* const_iterator::m_p_tbl` `[protected]`

Pointer to the table object which created the iterator (used for incrementing its position).

Definition at line 106 of file `unordered_iterator/const_iterator.hpp`.

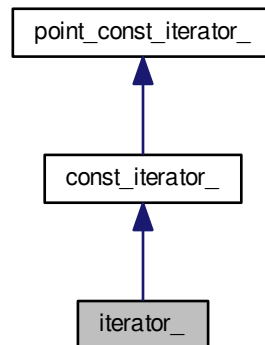
Referenced by `operator++()`, and `iterator_::operator++()`.

The documentation for this class was generated from the following file:

- [unordered\\_iterator/const\\_iterator.hpp](#)

#### 4.402 iterator\_ Class Reference

Inheritance diagram for `iterator_`:



### Public Types

- typedef const\_pointer\_ [const\\_pointer](#)
- typedef const\_reference\_ [const\\_reference](#)
- typedef \_Alloc::difference\_type [difference\\_type](#)
- typedef [std::forward\\_iterator\\_tag](#) iterator\_category
- typedef pointer\_ [pointer](#)
- typedef reference\_ [reference](#)
- typedef value\_type\_ [value\\_type](#)

### Public Member Functions

- [iterator\\_](#) ()
- [operator const point\\_iterator\\_](#) () const
- [operator point\\_iterator\\_](#) ()
- bool [operator!=](#) (const [point\\_iterator\\_](#) &other) const
- bool [operator!=](#) (const [point\\_const\\_iterator\\_](#) &other) const
- [reference operator\\*](#) () const
- [iterator\\_ & operator++](#) ()
- [iterator\\_ operator++](#) (int)
- [pointer operator->](#) () const
- bool [operator==](#) (const [point\\_iterator\\_](#) &other) const
- bool [operator==](#) (const [point\\_const\\_iterator\\_](#) &other) const

### Protected Types

- typedef [const\\_iterator\\_](#) **base\_type**

### Protected Member Functions

- [iterator\\_](#) ([pointer](#) p\_value, PB\_DS\_GEN\_POS pos, PB\_DS\_CLASS\_C\_DEC \*p\_tbl)

### Protected Attributes

- const PB\_DS\_CLASS\_C\_DEC \* [m\\_p\\_tbl](#)
- [const\\_pointer](#) **m\_p\_value**
- PB\_DS\_GEN\_POS **m\_pos**

### Friends

- class **PB\_DS\_CLASS\_C\_DEC**

#### 4.402.1 Detailed Description

Range-type iterator.

Definition at line 43 of file iterator.hpp.



#### 4.402.2 Member Typedef Documentation

##### 4.402.2.1 `typedef const_pointer_iterator_::const_pointer`

Iterator's const pointer type.

Definition at line 60 of file iterator.hpp.

##### 4.402.2.2 `typedef const_reference_iterator_::const_reference`

Iterator's const reference type.

Definition at line 66 of file iterator.hpp.

##### 4.402.2.3 `typedef _Alloc::difference_type iterator_::difference_type`

Difference type.

Definition at line 51 of file iterator.hpp.

##### 4.402.2.4 `typedef std::forward_iterator_tag iterator_::iterator_category`

Category.

Definition at line 48 of file iterator.hpp.

##### 4.402.2.5 `typedef pointer_iterator_::pointer`

Iterator's pointer type.

Definition at line 57 of file iterator.hpp.

##### 4.402.2.6 `typedef reference_iterator_::reference`

Iterator's reference type.

Definition at line 63 of file iterator.hpp.

##### 4.402.2.7 `typedef value_type_iterator_::value_type`

Iterator's value type.

Definition at line 54 of file iterator.hpp.

#### 4.402.3 Constructor & Destructor Documentation

##### 4.402.3.1 `iterator_::iterator_ ( ) [inline]`

Default constructor.

Definition at line 70 of file iterator.hpp.

##### 4.402.3.2 `iterator_::iterator_ ( pointer p_value, PB_DS_GEN_POS pos, PB_DS_CLASS_C_DEC * p_tbl ) [inline], [protected]`

Constructor used by the table to initiate the generalized pointer and position (e.g., this is called from within a find() of a table.

Definition at line 125 of file iterator.hpp.

## 4.402.4 Member Function Documentation

4.402.4.1 `iterator_::operator const point_iterator_ ( ) const` `[inline]`

Conversion to a point-type iterator.

Definition at line 80 of file iterator.hpp.

4.402.4.2 `iterator_::operator point_iterator_ ( )` `[inline]`

Conversion to a point-type iterator.

Definition at line 75 of file iterator.hpp.

4.402.4.3 `bool point_const_iterator_::operator!= ( const point_iterator_ & other ) const` `[inline]`, `[inherited]`

Compares content (negatively) to a different iterator object.

Definition at line 118 of file unordered\_iterator/point\_const\_iterator.hpp.

4.402.4.4 `bool point_const_iterator_::operator!= ( const point_const_iterator_ & other ) const` `[inline]`, `[inherited]`

Compares content (negatively) to a different iterator object.

Definition at line 123 of file unordered\_iterator/point\_const\_iterator.hpp.

4.402.4.5 `reference iterator_::operator* ( ) const` `[inline]`

Access.

Definition at line 93 of file iterator.hpp.

4.402.4.6 `iterator_ & iterator_::operator++ ( )` `[inline]`

Increments.

Definition at line 101 of file iterator.hpp.

References `const_iterator_::m_p_tbl`.

4.402.4.7 `iterator_ iterator_::operator++ ( int )` `[inline]`

Increments.

Definition at line 109 of file iterator.hpp.

References `const_iterator_::m_p_tbl`.

4.402.4.8 `pointer iterator_::operator-> ( ) const` `[inline]`

Access.

Definition at line 85 of file iterator.hpp.

4.402.4.9 `bool point_const_iterator_::operator== ( const point_iterator_ & other ) const` `[inline]`, `[inherited]`

Compares content to a different iterator object.

Definition at line 108 of file unordered\_iterator/point\_const\_iterator.hpp.

4.402.4.10 `bool point_const_iterator_::operator== ( const point_const_iterator_ & other ) const` `[inline]`,  
`[inherited]`

Compares content to a different iterator object.

Definition at line 113 of file `unordered_iterator/point_const_iterator.hpp`.

#### 4.402.5 Member Data Documentation

4.402.5.1 `const PB_DS_CLASS_C_DEC* const_iterator_::m_p_tbl` `[protected]`, `[inherited]`

Pointer to the table object which created the iterator (used for incrementing its position).

Definition at line 106 of file `unordered_iterator/const_iterator.hpp`.

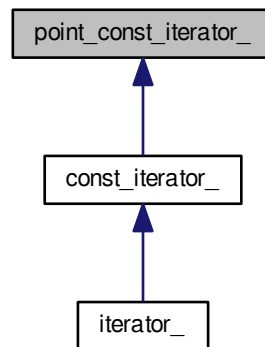
Referenced by `const_iterator_::operator++()`, and `operator++()`.

The documentation for this class was generated from the following file:

- [iterator.hpp](#)

#### 4.403 point\_const\_iterator\_ Class Reference

Inheritance diagram for `point_const_iterator_`:



#### Public Types

- `typedef const_pointer_ const_pointer`
- `typedef const_reference_ const_reference`
- `typedef trivial_iterator_difference_type difference_type`
- `typedef trivial_iterator_tag iterator_category`
- `typedef pointer_ pointer`
- `typedef reference_ reference`
- `typedef value_type_ value_type`

### Public Member Functions

- **point\_const\_iterator\_** ([const\\_pointer](#) p\_value)
- **point\_const\_iterator\_** ()
- **point\_const\_iterator\_** (const [point\\_const\\_iterator\\_](#) &other)
- **point\_const\_iterator\_** (const [point\\_iterator\\_](#) &other)
- bool **operator!=** (const [point\\_iterator\\_](#) &other) const
- bool **operator!=** (const [point\\_const\\_iterator\\_](#) &other) const
- **const\_reference** **operator\*** () const
- **const\_pointer** **operator->** () const
- bool **operator==** (const [point\\_iterator\\_](#) &other) const
- bool **operator==** (const [point\\_const\\_iterator\\_](#) &other) const

### Protected Attributes

- [const\\_pointer](#) **m\_p\_value**

### Friends

- class **PB\_DS\_CLASS\_C\_DEC**
- class **point\_iterator\_**

#### 4.403.1 Detailed Description

Const point-type iterator.

Definition at line 45 of file `unordered_iterator/point_const_iterator.hpp`.

#### 4.403.2 Member Typedef Documentation

##### 4.403.2.1 `typedef const_pointer_ point_const_iterator_::const_pointer`

Iterator's const pointer type.

Definition at line 61 of file `unordered_iterator/point_const_iterator.hpp`.

##### 4.403.2.2 `typedef const_reference_ point_const_iterator_::const_reference`

Iterator's const reference type.

Definition at line 67 of file `unordered_iterator/point_const_iterator.hpp`.

##### 4.403.2.3 `typedef trivial_iterator_difference_type point_const_iterator_::difference_type`

Difference type.

Definition at line 52 of file `unordered_iterator/point_const_iterator.hpp`.

##### 4.403.2.4 `typedef trivial_iterator_tag point_const_iterator_::iterator_category`

Category.

Definition at line 49 of file `unordered_iterator/point_const_iterator.hpp`.

#### 4.403.2.5 `typedef pointer_point_const_iterator::pointer`

Iterator's pointer type.

Definition at line 58 of file `unordered_iterator/point_const_iterator.hpp`.

#### 4.403.2.6 `typedef reference_point_const_iterator::reference`

Iterator's reference type.

Definition at line 64 of file `unordered_iterator/point_const_iterator.hpp`.

#### 4.403.2.7 `typedef value_type_point_const_iterator::value_type`

Iterator's value type.

Definition at line 55 of file `unordered_iterator/point_const_iterator.hpp`.

### 4.403.3 Constructor & Destructor Documentation

#### 4.403.3.1 `point_const_iterator::point_const_iterator( ) [inline]`

Default constructor.

Definition at line 75 of file `unordered_iterator/point_const_iterator.hpp`.

#### 4.403.3.2 `point_const_iterator::point_const_iterator( const point_const_iterator_ & other ) [inline]`

Copy constructor.

Definition at line 80 of file `unordered_iterator/point_const_iterator.hpp`.

#### 4.403.3.3 `point_const_iterator::point_const_iterator( const point_iterator_ & other ) [inline]`

Copy constructor.

Definition at line 86 of file `unordered_iterator/point_const_iterator.hpp`.

### 4.403.4 Member Function Documentation

#### 4.403.4.1 `bool point_const_iterator::operator!=( const point_iterator_ & other ) const [inline]`

Compares content (negatively) to a different iterator object.

Definition at line 118 of file `unordered_iterator/point_const_iterator.hpp`.

#### 4.403.4.2 `bool point_const_iterator::operator!=( const point_const_iterator_ & other ) const [inline]`

Compares content (negatively) to a different iterator object.

Definition at line 123 of file `unordered_iterator/point_const_iterator.hpp`.

#### 4.403.4.3 `const_reference point_const_iterator::operator*( ) const [inline]`

Access.

Definition at line 100 of file `unordered_iterator/point_const_iterator.hpp`.

4.403.4.4 `const_pointer point_const_iterator::operator-> ( ) const` `[inline]`

Access.

Definition at line 92 of file `unordered_iterator/point_const_iterator.hpp`.

4.403.4.5 `bool point_const_iterator::operator==( const point_iterator_ & other ) const` `[inline]`

Compares content to a different iterator object.

Definition at line 108 of file `unordered_iterator/point_const_iterator.hpp`.

4.403.4.6 `bool point_const_iterator::operator==( const point_const_iterator_ & other ) const` `[inline]`

Compares content to a different iterator object.

Definition at line 113 of file `unordered_iterator/point_const_iterator.hpp`.

The documentation for this class was generated from the following file:

- [unordered\\_iterator/point\\_const\\_iterator.hpp](#)

## 4.404 point\_iterator\_ Class Reference

### Public Types

- `typedef const_pointer_ const_pointer`
- `typedef const_reference_ const_reference`
- `typedef trivial_iterator_difference_type difference_type`
- `typedef trivial_iterator_tag iterator_category`
- `typedef pointer_ pointer`
- `typedef reference_ reference`
- `typedef value_type_ value_type`

### Public Member Functions

- `point_iterator_ ()`
- `point_iterator_ (const point_iterator_ &other)`
- `point_iterator_ (pointer p_value)`
- `bool operator!= (const point_iterator_ &other) const`
- `bool operator!= (const point_const_iterator_ &other) const`
- `reference operator* () const`
- `pointer operator-> () const`
- `bool operator==(const point_iterator_ &other) const`
- `bool operator==(const point_const_iterator_ &other) const`

### Protected Attributes

- `pointer m_p_value`

### Friends

- `class PB_DS_CLASS_C_DEC`
- `class point_const_iterator_`

#### 4.404.1 Detailed Description

Find type iterator.

Definition at line 43 of file point\_iterator.hpp.

#### 4.404.2 Member Typedef Documentation

##### 4.404.2.1 `typedef const_pointer_point_iterator::const_pointer`

Iterator's const pointer type.

Definition at line 59 of file point\_iterator.hpp.

##### 4.404.2.2 `typedef const_reference_point_iterator::const_reference`

Iterator's const reference type.

Definition at line 65 of file point\_iterator.hpp.

##### 4.404.2.3 `typedef trivial_iterator_difference_type_point_iterator::difference_type`

Difference type.

Definition at line 50 of file point\_iterator.hpp.

##### 4.404.2.4 `typedef trivial_iterator_tag_point_iterator::iterator_category`

Category.

Definition at line 47 of file point\_iterator.hpp.

##### 4.404.2.5 `typedef pointer_point_iterator::pointer`

Iterator's pointer type.

Definition at line 56 of file point\_iterator.hpp.

##### 4.404.2.6 `typedef reference_point_iterator::reference`

Iterator's reference type.

Definition at line 62 of file point\_iterator.hpp.

##### 4.404.2.7 `typedef value_type_point_iterator::value_type`

Iterator's value type.

Definition at line 53 of file point\_iterator.hpp.

#### 4.404.3 Constructor & Destructor Documentation

##### 4.404.3.1 `point_iterator::point_iterator( ) [inline]`

Default constructor.

Definition at line 69 of file point\_iterator.hpp.

## 4.404.3.2 point\_iterator::point\_iterator( const point\_iterator\_ &amp; other ) [inline]

Copy constructor.

Definition at line 75 of file point\_iterator.hpp.

## 4.404.4 Member Function Documentation

## 4.404.4.1 bool point\_iterator::operator!=( const point\_iterator\_ &amp; other ) const [inline]

Compares content to a different iterator object.

Definition at line 107 of file point\_iterator.hpp.

## 4.404.4.2 bool point\_iterator::operator!=( const point\_const\_iterator\_ &amp; other ) const [inline]

Compares content (negatively) to a different iterator object.

Definition at line 112 of file point\_iterator.hpp.

## 4.404.4.3 reference point\_iterator::operator\*( ) const [inline]

Access.

Definition at line 89 of file point\_iterator.hpp.

## 4.404.4.4 pointer point\_iterator::operator-&gt;( ) const [inline]

Access.

Definition at line 81 of file point\_iterator.hpp.

## 4.404.4.5 bool point\_iterator::operator==( const point\_iterator\_ &amp; other ) const [inline]

Compares content to a different iterator object.

Definition at line 97 of file point\_iterator.hpp.

## 4.404.4.6 bool point\_iterator::operator==( const point\_const\_iterator\_ &amp; other ) const [inline]

Compares content to a different iterator object.

Definition at line 102 of file point\_iterator.hpp.

The documentation for this class was generated from the following file:

- [point\\_iterator.hpp](#)

## 4.405 std::\_\_atomic\_base&lt;\_ITp&gt; Struct Template Reference

## Public Member Functions

- **\_\_atomic\_base** (const [\\_\\_atomic\\_base](#) &)=delete
- constexpr **\_\_atomic\_base** (\_\_int\_type \_\_i) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept



- `bool compare_exchange_strong ( __int_type & __i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept`
- `bool compare_exchange_strong ( __int_type & __i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `bool compare_exchange_weak ( __int_type & __i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept`
- `bool compare_exchange_weak ( __int_type & __i1, __int_type __i2, memory_order __m1, memory_order __m2) volatile noexcept`
- `bool compare_exchange_weak ( __int_type & __i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept`
- `bool compare_exchange_weak ( __int_type & __i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `__int_type exchange ( __int_type __i, memory_order __m=memory_order_seq_cst) noexcept`
- `__int_type exchange ( __int_type __i, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `__int_type fetch_add ( __int_type __i, memory_order __m=memory_order_seq_cst) noexcept`
- `__int_type fetch_add ( __int_type __i, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `__int_type fetch_and ( __int_type __i, memory_order __m=memory_order_seq_cst) noexcept`
- `__int_type fetch_and ( __int_type __i, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `__int_type fetch_or ( __int_type __i, memory_order __m=memory_order_seq_cst) noexcept`
- `__int_type fetch_or ( __int_type __i, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `__int_type fetch_sub ( __int_type __i, memory_order __m=memory_order_seq_cst) noexcept`
- `__int_type fetch_sub ( __int_type __i, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `__int_type fetch_xor ( __int_type __i, memory_order __m=memory_order_seq_cst) noexcept`
- `__int_type fetch_xor ( __int_type __i, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `bool is_lock_free () const noexcept`
- `bool is_lock_free () const volatile noexcept`
- `__int_type load (memory_order __m=memory_order_seq_cst) const noexcept`
- `__int_type load (memory_order __m=memory_order_seq_cst) const volatile noexcept`
- `operator __int_type () const noexcept`
- `operator __int_type () const volatile noexcept`
- `__int_type operator&= ( __int_type __i) noexcept`
- `__int_type operator&= ( __int_type __i) volatile noexcept`
- `__int_type operator++ (int) noexcept`
- `__int_type operator++ (int) volatile noexcept`
- `__int_type operator++ () noexcept`
- `__int_type operator++ () volatile noexcept`
- `__int_type operator+= ( __int_type __i) noexcept`
- `__int_type operator+= ( __int_type __i) volatile noexcept`
- `__int_type operator-- (int) noexcept`
- `__int_type operator-- (int) volatile noexcept`
- `__int_type operator-- () noexcept`
- `__int_type operator-- () volatile noexcept`
- `__int_type operator-= ( __int_type __i) noexcept`
- `__int_type operator-= ( __int_type __i) volatile noexcept`
- `__atomic_base & operator= (const __atomic_base &)=delete`
- `__atomic_base & operator= (const __atomic_base &) volatile=delete`
- `__int_type operator= ( __int_type __i) noexcept`
- `__int_type operator= ( __int_type __i) volatile noexcept`
- `__int_type operator^= ( __int_type __i) noexcept`
- `__int_type operator^= ( __int_type __i) volatile noexcept`
- `__int_type operator|= ( __int_type __i) noexcept`
- `__int_type operator|= ( __int_type __i) volatile noexcept`
- `void store ( __int_type __i, memory_order __m=memory_order_seq_cst) noexcept`
- `void store ( __int_type __i, memory_order __m=memory_order_seq_cst) volatile noexcept`

## 4.405.1 Detailed Description

```
template<typename _ITp>struct std::__atomic_base<_ITp>
```

Base class for atomic integrals.

Definition at line 117 of file atomic\_base.h.

The documentation for this struct was generated from the following file:

- [atomic\\_base.h](#)

## 4.406 std::\_\_atomic\_base&lt;\_PTp\*&gt; Struct Template Reference

## Public Member Functions

- **\_\_atomic\_base** (const [\\_\\_atomic\\_base](#) &)=delete
- constexpr **\_\_atomic\_base** ([\\_\\_pointer\\_type](#) \_\_p) noexcept
- bool **compare\_exchange\_strong** ([\\_\\_pointer\\_type](#) &\_\_p1, [\\_\\_pointer\\_type](#) \_\_p2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** ([\\_\\_pointer\\_type](#) &\_\_p1, [\\_\\_pointer\\_type](#) \_\_p2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- [\\_\\_pointer\\_type](#) **exchange** ([\\_\\_pointer\\_type](#) \_\_p, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) noexcept
- [\\_\\_pointer\\_type](#) **exchange** ([\\_\\_pointer\\_type](#) \_\_p, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) volatile noexcept
- [\\_\\_pointer\\_type](#) **fetch\_add** (ptrdiff\_t \_\_d, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) noexcept
- [\\_\\_pointer\\_type](#) **fetch\_add** (ptrdiff\_t \_\_d, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) volatile noexcept
- [\\_\\_pointer\\_type](#) **fetch\_sub** (ptrdiff\_t \_\_d, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) noexcept
- [\\_\\_pointer\\_type](#) **fetch\_sub** (ptrdiff\_t \_\_d, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- [\\_\\_pointer\\_type](#) **load** ([memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) const noexcept
- [\\_\\_pointer\\_type](#) **load** ([memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) const volatile noexcept
- **operator** [\\_\\_pointer\\_type](#) () const noexcept
- **operator** [\\_\\_pointer\\_type](#) () const volatile noexcept
- [\\_\\_pointer\\_type](#) **operator++** (int) noexcept
- [\\_\\_pointer\\_type](#) **operator++** (int) volatile noexcept
- [\\_\\_pointer\\_type](#) **operator++** () noexcept
- [\\_\\_pointer\\_type](#) **operator++** () volatile noexcept
- [\\_\\_pointer\\_type](#) **operator+=** (ptrdiff\_t \_\_d) noexcept
- [\\_\\_pointer\\_type](#) **operator+=** (ptrdiff\_t \_\_d) volatile noexcept
- [\\_\\_pointer\\_type](#) **operator--** (int) noexcept
- [\\_\\_pointer\\_type](#) **operator--** (int) volatile noexcept
- [\\_\\_pointer\\_type](#) **operator--** () noexcept
- [\\_\\_pointer\\_type](#) **operator--** () volatile noexcept
- [\\_\\_pointer\\_type](#) **operator-=** (ptrdiff\_t \_\_d) noexcept
- [\\_\\_pointer\\_type](#) **operator-=** (ptrdiff\_t \_\_d) volatile noexcept
- [\\_\\_atomic\\_base](#) & **operator=** (const [\\_\\_atomic\\_base](#) &)=delete
- [\\_\\_atomic\\_base](#) & **operator=** (const [\\_\\_atomic\\_base](#) &) volatile=delete
- [\\_\\_pointer\\_type](#) **operator=** ([\\_\\_pointer\\_type](#) \_\_p) noexcept
- [\\_\\_pointer\\_type](#) **operator=** ([\\_\\_pointer\\_type](#) \_\_p) volatile noexcept
- void **store** ([\\_\\_pointer\\_type](#) \_\_p, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) noexcept
- void **store** ([\\_\\_pointer\\_type](#) \_\_p, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) volatile noexcept

#### 4.406.1 Detailed Description

```
template<typename _PTp>struct std::__atomic_base<_PTp * >
```

Partial specialization for pointer types.

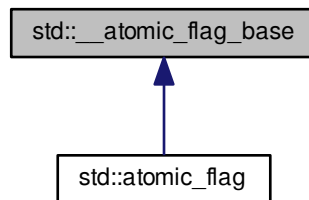
Definition at line 665 of file atomic\_base.h.

The documentation for this struct was generated from the following file:

- [atomic\\_base.h](#)

#### 4.407 std::\_\_atomic\_flag\_base Struct Reference

Inheritance diagram for std::\_\_atomic\_flag\_base:



##### Public Attributes

- `__atomic_flag_data_type _M_i`

#### 4.407.1 Detailed Description

Base type for atomic\_flag.

Base type is POD with data, allowing atomic\_flag to derive from it and meet the standard layout type requirement. In addition to compatibility with a C interface, this allows different implementations of atomic\_flag to use the same atomic operation functions, via a standard conversion to the \_\_atomic\_flag\_base argument.

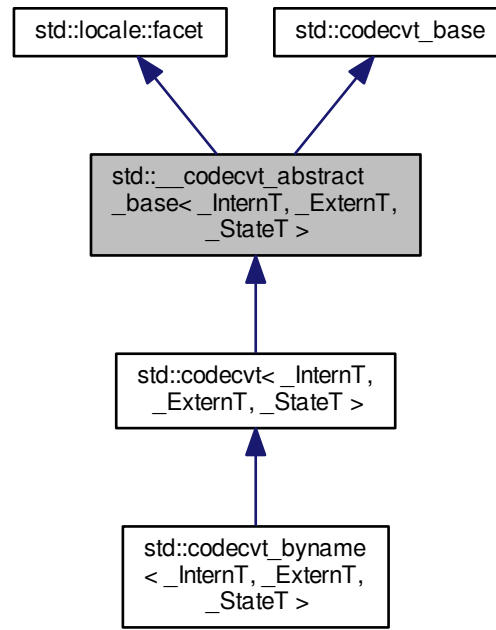
Definition at line 261 of file atomic\_base.h.

The documentation for this struct was generated from the following file:

- [atomic\\_base.h](#)

## 4.408 std::\_\_codecvt\_abstract\_base&lt; \_InternT, \_ExternT, \_StateT &gt; Class Template Reference

Inheritance diagram for std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, \_StateT >:



## Public Types

- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state\_type**

## Public Member Functions

- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

## Protected Member Functions

- **\_\_codecvt\_abstract\_base** (size\_t \_\_refs=0)
- virtual bool **do\_always\_noconv** () const =0 throw ()
- virtual int **do\_encoding** () const =0 throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \* \_\_from, const extern\_type \* \_\_from\_end, const extern\_type \*& \_\_from\_next, intern\_type \* \_\_to, intern\_type \* \_\_to\_end, intern\_type \*& \_\_to\_next) const =0
- virtual int **do\_length** (state\_type &, const extern\_type \* \_\_from, const extern\_type \* \_\_end, size\_t \_\_max) const =0
- virtual int **do\_max\_length** () const =0 throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \* \_\_from, const intern\_type \* \_\_from\_end, const intern\_type \*& \_\_from\_next, extern\_type \* \_\_to, extern\_type \* \_\_to\_end, extern\_type \*& \_\_to\_next) const =0
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \* \_\_to, extern\_type \* \_\_to\_end, extern\_type \*& \_\_to↔\_next) const =0

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \* \_\_s)

## 4.408.1 Detailed Description

```
template<typename _InternT, typename _ExternT, typename _StateT>class std::__codecvt_abstract_base< _InternT, _ExternT, _↔
StateT >
```

Common base for codecvt functions.

This template class provides implementations of the public functions that forward to the protected virtual functions.

This template also provides abstract stubs for the protected virtual functions.

Definition at line 68 of file codecvt.h.

## 4.408.2 Member Function Documentation

4.408.2.1 `template<typename _InternT, typename _ExternT, typename _StateT> virtual result std::__codecvt_abstract_↔  
base< _InternT, _ExternT, _StateT >::do_out ( state_type & __state, const intern_type * __from, const intern_type *  
__from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next )  
const [protected],[pure virtual]`

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implemented in [std::codecvt< wchar\\_t, char, mbstate\\_t >](#), [std::codecvt< char, char, mbstate\\_t >](#), [std::codecvt< \\_↔\\_InternT, \\_ExternT, \\_StateT >](#), and [std::codecvt< \\_InternT, \\_ExternT, encoding\\_state >](#).

Referenced by [std::\\_\\_codecvt\\_abstract\\_base< \\_InternT, \\_ExternT, encoding\\_state >::out\(\)](#).

```
4.408.2.2 template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base<
    _InternT, _ExternT, _StateT >::in ( state_type & __state, const extern_type * __from, const extern_type * __from_end,
    const extern_type *& __from_next, intern_type * __to, intern_type * __to_end, intern_type *& __to_next ) const
    [inline]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

#### Returns

`codecvt_base::result`.

Definition at line 196 of file `codecvt.h`.

```
4.408.2.3 template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base<
    _InternT, _ExternT, _StateT >::out ( state_type & __state, const intern_type * __from, const intern_type * __from_end,
    const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next ) const
    [inline]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the

character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

#### Returns

`codecvt_base::result`.

Definition at line 116 of file `codecvt.h`.

```
4.408.2.4 template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base<
    _InternT, _ExternT, _StateT>::unshift ( state_type & __state, extern_type * __to, extern_type * __to_end, extern_type
    *& __to_next ) const [inline]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

#### Parameters

<code>__state</code>	Persistent conversion state data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

#### Returns

`codecvt_base::result`.

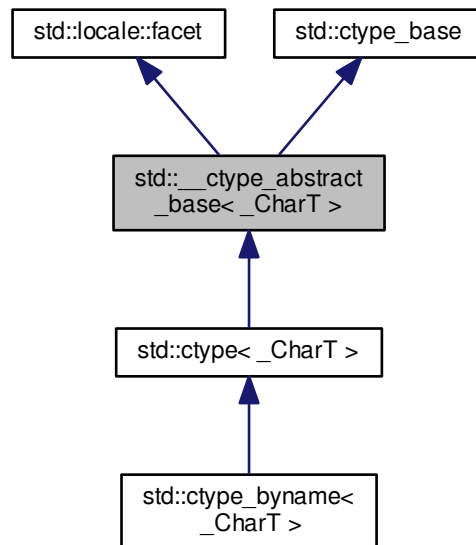
Definition at line 155 of file `codecvt.h`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 4.409 std::\_\_ctype\_abstract\_base&lt;\_CharT&gt; Class Template Reference

Inheritance diagram for std::\_\_ctype\_abstract\_base<\_CharT>:



## Public Types

- typedef const int \* **\_\_to\_type**
- typedef \_CharT **char\_type**
- typedef unsigned short **mask**

## Public Member Functions

- bool **is** (mask \_\_m, **char\_type** \_\_c) const
- const **char\_type** \* **is** (const **char\_type** \* \_\_lo, const **char\_type** \* \_\_hi, mask \*\_\_vec) const
- char **narrow** (**char\_type** \_\_c, char \_\_default) const
- const **char\_type** \* **narrow** (const **char\_type** \* \_\_lo, const **char\_type** \* \_\_hi, char \_\_default, char \*\_\_to) const
- const **char\_type** \* **scan\_is** (mask \_\_m, const **char\_type** \* \_\_lo, const **char\_type** \* \_\_hi) const
- const **char\_type** \* **scan\_not** (mask \_\_m, const **char\_type** \* \_\_lo, const **char\_type** \* \_\_hi) const
- **char\_type** **tolower** (**char\_type** \_\_c) const
- const **char\_type** \* **tolower** (**char\_type** \* \_\_lo, const **char\_type** \* \_\_hi) const
- **char\_type** **toupper** (**char\_type** \_\_c) const
- const **char\_type** \* **toupper** (**char\_type** \* \_\_lo, const **char\_type** \* \_\_hi) const
- **char\_type** **widen** (char \_\_c) const
- const char \* **widen** (const char \* \_\_lo, const char \* \_\_hi, **char\_type** \* \_\_to) const



### Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

### Protected Member Functions

- **\_\_ctype\_abstract\_base** (size\_t \_\_refs=0)
- virtual bool **do\_is** (mask \_\_m, char\_type \_\_c) const =0
- virtual const char\_type \* **do\_is** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, mask \* \_\_vec) const =0
- virtual char **do\_narrow** (char\_type \_\_c, char \_\_dfault) const =0
- virtual const char\_type \* **do\_narrow** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_dfault, char \* \_\_to) const =0
- virtual const char\_type \* **do\_scan\_is** (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const =0
- virtual const char\_type \* **do\_scan\_not** (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const =0
- virtual char\_type **do\_tolower** (char\_type \_\_c) const =0
- virtual const char\_type \* **do\_tolower** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const =0
- virtual char\_type **do\_toupper** (char\_type \_\_c) const =0
- virtual const char\_type \* **do\_toupper** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const =0
- virtual char\_type **do\_widen** (char \_\_c) const =0
- virtual const char \* **do\_widen** (const char \* \_\_lo, const char \* \_\_hi, char\_type \* \_\_to) const =0

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale & \_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale & \_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale & \_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \* \_\_s)

#### 4.409.1 Detailed Description

```
template<typename _CharT>class std::__ctype_abstract_base<_CharT>
```

Common base for ctype facet.

This template class provides implementations of the public functions that forward to the protected virtual functions.

This template also provides abstract stubs for the protected virtual functions.

Definition at line 143 of file locale\_facets.h.

## 4.409.2 Member Typedef Documentation

## 4.409.2.1 template&lt;typename \_CharT&gt; typedef \_CharT std::\_\_ctype\_abstract\_base&lt;\_CharT&gt;::char\_type

Typedef for the template parameter.

Definition at line 148 of file locale\_facets.h.

## 4.409.3 Member Function Documentation

## 4.409.3.1 template&lt;typename \_CharT&gt; virtual bool std::\_\_ctype\_abstract\_base&lt;\_CharT&gt;::do\_is ( mask \_\_m, char\_type \_\_c ) const [protected], [pure virtual]

Test char\_type classification.

This function finds a mask M for c and compares it to mask m.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

## Parameters

<code>__c</code>	The char_type to find the mask of.
<code>__m</code>	The mask to compare against.

## Returns

$(M \& \text{__m}) \neq 0$ .

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

Referenced by [std::\\_\\_ctype\\_abstract\\_base<wchar\\_t>::is\(\)](#).

## 4.409.3.2 template&lt;typename \_CharT&gt; virtual const char\_type\* std::\_\_ctype\_abstract\_base&lt;\_CharT&gt;::do\_is ( const char\_type \* \_\_lo, const char\_type \* \_\_hi, mask \* \_\_vec ) const [protected], [pure virtual]

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.

## Returns

`__hi`.

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

## 4.409.3.3 template&lt;typename \_CharT&gt; virtual char std::\_\_ctype\_abstract\_base&lt;\_CharT&gt;::do\_narrow ( char\_type \_\_c, char \_\_dfault ) const [protected], [pure virtual]

Narrow char\_type to char.

This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

<code>__c</code>	The <code>char_type</code> to convert.
<code>__dfault</code>	Char to return if conversion fails.

#### Returns

The converted char.

Implemented in `std::ctype< wchar_t >`, and `std::ctype< _CharT >`.

Referenced by `std::__ctype_abstract_base< wchar_t >::narrow()`.

4.409.3.4 `template<typename _CharT> virtual const char_type* std::__ctype_abstract_base< _CharT >::do_narrow (const char_type * __lo, const char_type * __hi, char __dfault, char * __to ) const [protected], [pure virtual]`

Narrow `char_type` array to char.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `__dfault` is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__dfault</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

#### Returns

`__hi`.

Implemented in `std::ctype< wchar_t >`, and `std::ctype< _CharT >`.

4.409.3.5 `template<typename _CharT> virtual const char_type* std::__ctype_abstract_base< _CharT >::do_scan_is (mask __m, const char_type * __lo, const char_type * __hi ) const [protected], [pure virtual]`

Find `char_type` matching mask.

This function searches for and returns the first `char_type` `c` in `[__lo,__hi)` for which `is(__m,c)` is true.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

## Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

## Returns

Pointer to a matching `char_type` if found, else `__hi`.

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

Referenced by `std::__ctype_abstract_base<wchar_t>::scan_is()`.

4.409.3.6 `template<typename _CharT> virtual const char_type* std::__ctype_abstract_base<_CharT>::do_scan_not ( mask __m, const char_type * __lo, const char_type * __hi ) const` `[protected], [pure virtual]`

Find `char_type` not matching mask.

This function searches for and returns a pointer to the first `char_type` `c` of `[lo,hi)` for which `is(m,c)` is false.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

## Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

## Returns

Pointer to a non-matching `char_type` if found, else `__hi`.

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

Referenced by `std::__ctype_abstract_base<wchar_t>::scan_not()`.

4.409.3.7 `template<typename _CharT> virtual char_type std::__ctype_abstract_base<_CharT>::do_tolower ( char_type __c ) const` `[protected], [pure virtual]`

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

## Parameters

<code>__c</code>	The <code>char_type</code> to convert.
------------------	--

## Returns

The lowercase `char_type` if convertible, else `__c`.

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

Referenced by `std::__ctype_abstract_base<wchar_t>::tolower()`.

**4.409.3.8** `template<typename _CharT> virtual const char_type* std::__ctype_abstract_base<_CharT>::do_tolower (char_type * __lo, const char_type * __hi) const` [protected], [pure virtual]

Convert array to lowercase.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

#### Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

#### Returns

`__hi`.

Implemented in `std::ctype<wchar_t>`, and `std::ctype<_CharT>`.

**4.409.3.9** `template<typename _CharT> virtual char_type std::__ctype_abstract_base<_CharT>::do_toupper (char_type __c) const` [protected], [pure virtual]

Convert to uppercase.

This virtual function converts the `char_type` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

#### Parameters

<code>__c</code>	The <code>char_type</code> to convert.
------------------	--

#### Returns

The uppercase `char_type` if convertible, else `__c`.

Implemented in `std::ctype<wchar_t>`, and `std::ctype<_CharT>`.

Referenced by `std::__ctype_abstract_base<wchar_t>::toupper()`.

**4.409.3.10** `template<typename _CharT> virtual const char_type* std::__ctype_abstract_base<_CharT>::do_toupper (char_type * __lo, const char_type * __hi) const` [protected], [pure virtual]

Convert array to uppercase.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to uppercase if possible. Other elements remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

## Returns

`__hi`.

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

**4.409.3.11** `template<typename _CharT> virtual char_type std::__ctype_abstract_base<_CharT>::do_widen ( char __c )  
const [protected], [pure virtual]`

Widen char.

This virtual function converts the char to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

## Returns

The converted char\_type

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

Referenced by `std::__ctype_abstract_base<wchar_t>::widen()`.

**4.409.3.12** `template<typename _CharT> virtual const char* std::__ctype_abstract_base<_CharT>::do_widen ( const char  
* __lo, const char * __hi, char_type * __to ) const [protected], [pure virtual]`

Widen char array.

This function converts each char in the input to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

<code>__lo</code>	Pointer to start range.
<code>__hi</code>	Pointer to end of range.
<code>__to</code>	Pointer to the destination array.

## Returns

`__hi`.

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

**4.409.3.13** `template<typename _CharT> bool std::__ctype_abstract_base<_CharT>::is ( mask __m, char_type __c )  
const [inline]`

Test char\_type classification.

This function finds a mask M for \_\_c and compares it to mask \_\_m. It does so by returning the value of ctype<char\_type>::do\_is().

#### Parameters

<code>__c</code>	The char_type to compare the mask of.
<code>__m</code>	The mask to compare against.

#### Returns

$(M \& \text{__m}) \neq 0$ .

Definition at line 162 of file locale\_facets.h.

**4.409.3.14** `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::is ( const char_type  
* __lo, const char_type * __hi, mask * __vec ) const [inline]`

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char\_type>::do\_is().

#### Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.

#### Returns

`__hi`.

Definition at line 179 of file locale\_facets.h.

**4.409.3.15** `template<typename _CharT> char std::__ctype_abstract_base<_CharT>::narrow ( char_type __c, char  
__dfault ) const [inline]`

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_c).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

<code>__c</code>	The char_type to convert.
<code>__dfault</code>	Char to return if conversion fails.

#### Returns

The converted char.

Definition at line 324 of file locale\_facets.h.

4.409.3.16 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::narrow ( const char_type * __lo, const char_type * __hi, char __dfault, char * __to ) const [inline]`

Narrow array to char array.

This function converts each `char_type` in the input to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `char_type` in the input that cannot be converted, *dfault* is used instead. It does so by returning `ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__dfault</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

#### Returns

`__hi`.

Definition at line 346 of file `locale_facets.h`.

4.409.3.17 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::scan_is ( mask __m, const char_type * __lo, const char_type * __hi ) const [inline]`

Find `char_type` matching a mask.

This function searches for and returns the first `char_type` `c` in `[lo,hi)` for which `is(m,c)` is true. It does so by returning `ctype<char_type>::do_scan_is()`.

#### Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

#### Returns

Pointer to matching `char_type` if found, else `__hi`.

Definition at line 195 of file `locale_facets.h`.

4.409.3.18 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::scan_not ( mask __m, const char_type * __lo, const char_type * __hi ) const [inline]`

Find `char_type` not matching a mask.

This function searches for and returns the first `char_type` `c` in `[lo,hi)` for which `is(m,c)` is false. It does so by returning `ctype<char_type>::do_scan_not()`.

#### Parameters

<code>__m</code>	The mask to compare against.
------------------	------------------------------



<code>__lo</code>	Pointer to first char in range.
<code>__hi</code>	Pointer to end of range.

**Returns**

Pointer to non-matching char if found, else `__hi`.

Definition at line 211 of file `locale_facets.h`.

```
4.409.3.19 template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::tolower ( char_type __c )
          const [inline]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_tolower(c)`.

**Parameters**

<code>__c</code>	The <code>char_type</code> to convert.
------------------	--

**Returns**

The lowercase `char_type` if convertible, else `__c`.

Definition at line 254 of file `locale_facets.h`.

```
4.409.3.20 template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::tolower ( char_type
          * __lo, const char_type * __hi ) const [inline]
```

Convert array to lowercase.

This function converts each `char_type` in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_tolower(__lo, __hi)`.

**Parameters**

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

**Returns**

`__hi`.

Definition at line 269 of file `locale_facets.h`.

```
4.409.3.21 template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::toupper ( char_type __c )
          const [inline]
```

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

## Parameters

<code>__c</code>	The char_type to convert.
------------------	---------------------------

## Returns

The uppercase char\_type if convertible, else `__c`.

Definition at line 225 of file locale\_facets.h.

**4.409.3.22** `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::toupper ( char_type * __lo, const char_type * __hi ) const [inline]`

Convert array to uppercase.

This function converts each char\_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

## Returns

`__hi`.

Definition at line 240 of file locale\_facets.h.

**4.409.3.23** `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::widen ( char __c ) const [inline]`

Widen char to char\_type.

This function converts the char argument to char\_type using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

## Returns

The converted char\_type.

Definition at line 286 of file locale\_facets.h.

**4.409.3.24** `template<typename _CharT> const char* std::__ctype_abstract_base<_CharT>::widen ( const char * __lo, const char * __hi, char_type * __to ) const [inline]`

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__to</code>	Pointer to the destination array.

## Returns

`__hi`.

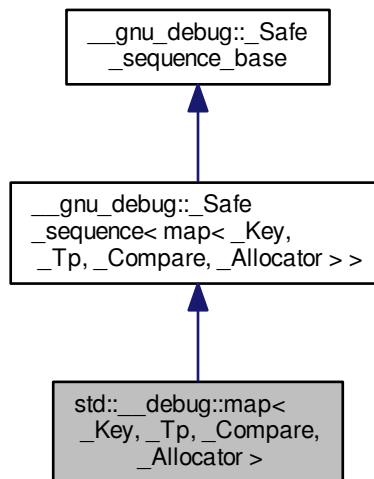
Definition at line 305 of file `locale_facets.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

#### 4.410 `std::__debug::map<_Key, _Tp, _Compare, _Allocator>` Class Template Reference

Inheritance diagram for `std::__debug::map<_Key, _Tp, _Compare, _Allocator>`:



## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::_Safe_iterator<_Base_const_iterator, map>` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator<const_iterator>` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::_Safe_iterator<_Base_iterator, map>` **iterator**

- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `std::pair< const _Key, _Tp >` **value\_type**

#### Public Member Functions

- **map** (const `_Compare` &\_\_comp=\_Compare(), const `_Allocator` &\_\_a=\_Allocator())
- template<typename `_InputIterator` > **map** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp=\_\_comp(), const `_Allocator` &\_\_a=\_Allocator())
- **map** (const `map` &\_\_x)
- **map** (const `_Base` &\_\_x)
- **map** (`map` &&\_\_x) noexcept(is\_nothrow\_copy\_constructible< `_Compare` >::value)
- **map** (initializer\_list< `value_type` > \_\_l, const `_Compare` &\_\_c=\_Compare(), const `allocator_type` &\_\_a=\_\_a(), const `allocator_type` &\_\_a=\_\_a())
- void **\_M\_attach** (`_Safe_iterator_base` \*\_\_it, bool \_\_constant)
- void **\_M\_attach\_single** (`_Safe_iterator_base` \*\_\_it, bool \_\_constant) throw ()
- `_Base` & **\_M\_base** () noexcept
- const `_Base` & **\_M\_base** () const noexcept
- void **\_M\_detach** (`_Safe_iterator_base` \*\_\_it)
- void **\_M\_detach\_single** (`_Safe_iterator_base` \*\_\_it) throw ()
- void **\_M\_invalidate\_all** () const
- void **\_M\_invalidate\_if** (`_Predicate` \_\_pred)
- void **\_M\_transfer\_from\_if** (`_Safe_sequence` &\_\_from, `_Predicate` \_\_pred)
- `iterator` **begin** () noexcept
- const `iterator` **begin** () const noexcept
- const `iterator` **cbegin** () const noexcept
- const `iterator` **cend** () const noexcept
- void **clear** () noexcept
- const `reverse_iterator` **crbegin** () const noexcept
- const `reverse_iterator` **crend** () const noexcept
- template<typename... `_Args`> `std::pair< iterator, bool >` **emplace** (`_Args` &&... \_\_args)
- template<typename... `_Args`> `iterator` **emplace\_hint** (const `iterator` \_\_pos, `_Args` &&... \_\_args)
- `iterator` **end** () noexcept
- const `iterator` **end** () const noexcept
- `std::pair< iterator, iterator >` **equal\_range** (const `key_type` &\_\_x)
- `std::pair< const_iterator, const_iterator >` **equal\_range** (const `key_type` &\_\_x) const
- `iterator` **erase** (const `iterator` \_\_position)
- `iterator` **erase** (`iterator` \_\_position)
- `size_type` **erase** (const `key_type` &\_\_x)
- `iterator` **erase** (const `iterator` \_\_first, const `iterator` \_\_last)
- `iterator` **find** (const `key_type` &\_\_x)
- const `iterator` **find** (const `key_type` &\_\_x) const
- `std::pair< iterator, bool >` **insert** (const `value_type` &\_\_x)
- template<typename `_Pair` , typename = typename std::enable\_if<std::is\_constructible<value\_type, `_Pair`&&>::value>::type> `std::pair< iterator, bool >` **insert** (`_Pair` &&\_\_x)

- void **insert** (std::initializer\_list< [value\\_type](#) > \_\_list)
- [iterator](#) **insert** (const [iterator](#) \_\_position, const [value\\_type](#) &\_\_x)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<[value\\_type](#), \_Pair&&>::value>::type> [iterator](#) **insert** (const [iterator](#) \_\_position, \_Pair &&\_\_x)
- template<typename \_InputIterator > void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [iterator](#) **lower\_bound** (const key\_type &\_\_x)
- const [iterator](#) **lower\_bound** (const key\_type &\_\_x) const
- [map](#) & **operator=** (const [map](#) &\_\_x)
- [map](#) & **operator=** ([map](#) &&\_\_x)
- [map](#) & **operator=** (initializer\_list< [value\\_type](#) > \_\_l)
- [reverse\\_iterator](#) **rbegin** () noexcept
- const [reverse\\_iterator](#) **rbegin** () const noexcept
- [reverse\\_iterator](#) **rend** () noexcept
- const [reverse\\_iterator](#) **rend** () const noexcept
- void **swap** ([map](#) &\_\_x)
- [iterator](#) **upper\_bound** (const key\_type &\_\_x)
- const [iterator](#) **upper\_bound** (const key\_type &\_\_x) const

#### Public Attributes

- \_Safe\_iterator\_base \* [\\_M\\_const\\_iterators](#)
- \_Safe\_iterator\_base \* [\\_M\\_iterators](#)
- unsigned int [\\_M\\_version](#)

#### Protected Member Functions

- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- \_\_gnu\_cxx::\_\_mutex & [\\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_revalidate\\_singular](#) ()
- void [\\_M\\_swap](#) (\_Safe\_sequence\_base &\_\_x)

#### 4.410.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std::pair<const _Key, _Tp> >>> class std::__debug::map< _Key, _Tp, _Compare, _Allocator >
```

Class std::map with safety/checking/debug instrumentation.

Definition at line 43 of file debug/map.h.

#### 4.410.2 Member Function Documentation

4.410.2.1 void \_\_gnu\_debug::Safe\_sequence\_base::M\_attach ( \_Safe\_iterator\_base \* \_\_it, bool \_\_constant )  
[inherited]

Attach an iterator to this sequence.

4.410.2.2 void \_\_gnu\_debug::Safe\_sequence\_base::M\_attach\_single ( \_Safe\_iterator\_base \* \_\_it, bool \_\_constant ) throw()  
[inherited]

Likewise but not thread safe.

4.410.2.3 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach ( \_Safe\_iterator\_base \* \_\_it ) [inherited]

Detach an iterator from this sequence

4.410.2.4 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_all ( ) [protected], [inherited]

Detach all iterators, leaving them singular.

Referenced by \_\_gnu\_debug::Safe\_sequence\_base::~~Safe\_sequence\_base().

4.410.2.5 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_single ( \_Safe\_iterator\_base \* \_\_it ) throw ( ) [inherited]

Likewise but not thread safe.

4.410.2.6 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_singular ( ) [protected], [inherited]

Detach all singular iterators.

#### Postcondition

for all iterators i attached to this sequence, i->\_M\_version == \_M\_version.

4.410.2.7 \_\_gnu\_cxx::mutex& \_\_gnu\_debug::Safe\_sequence\_base::M\_get\_mutex ( ) throw ( ) [protected], [inherited]

For use in \_Safe\_sequence.

4.410.2.8 void \_\_gnu\_debug::Safe\_sequence\_base::M\_invalidate\_all ( ) const [inline], [inherited]

Invalidates all iterators.

Definition at line 233 of file safe\_base.h.

4.410.2.9 void \_\_gnu\_debug::Safe\_sequence< map<\_Key, \_Tp, \_Compare, \_Allocator > >::M\_invalidate\_if ( \_Predicate \_\_pred ) [inherited]

Invalidates all iterators x that reference this sequence, are not singular, and for which \_\_pred(x) returns true. \_\_pred will be invoked with the normal iterators nested in the safe ones.

4.410.2.10 void \_\_gnu\_debug::Safe\_sequence\_base::M\_revalidate\_singular ( ) [protected], [inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.410.2.11 void \_\_gnu\_debug::Safe\_sequence\_base::M\_swap ( \_Safe\_sequence\_base & \_\_x ) [protected], [inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.410.2.12 void \_\_gnu\_debug::Safe\_sequence< map<\_Key, \_Tp, \_Compare, \_Allocator > >::M\_transfer\_from\_if ( \_Safe\_sequence< map<\_Key, \_Tp, \_Compare, \_Allocator > > & \_\_from, \_Predicate \_\_pred ) [inherited]

Transfers all iterators x that reference from sequence, are not singular, and for which \_\_pred(x) returns true. \_\_pred will be invoked with the normal iterators nested in the safe ones.

#### 4.410.3 Member Data Documentation

##### 4.410.3.1 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators` [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file `safe_base.h`.

##### 4.410.3.2 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file `safe_base.h`.

##### 4.410.3.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable],[inherited]

The container version number. This number may never be 0.

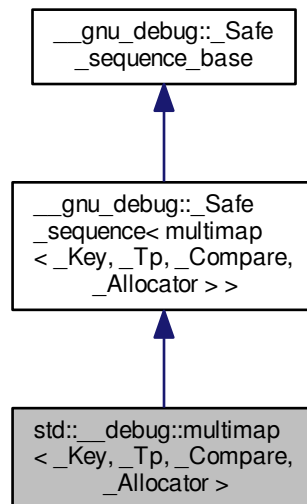
Definition at line 187 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [debug/map.h](#)

#### 4.411 `std::__debug::multimap<_Key,_Tp,_Compare,_Allocator>` Class Template Reference

Inheritance diagram for `std::__debug::multimap<_Key,_Tp,_Compare,_Allocator>`:



#### Public Types

- typedef `_Allocator` **allocator\_type**

- typedef `__gnu_debug::_Safe_iterator<_Base_const_iterator, multimap>` `const_iterator`
- typedef `_Base::const_pointer` `const_pointer`
- typedef `_Base::const_reference` `const_reference`
- typedef `std::reverse_iterator<const_iterator>` `const_reverse_iterator`
- typedef `_Base::difference_type` `difference_type`
- typedef `__gnu_debug::_Safe_iterator<_Base_iterator, multimap>` `iterator`
- typedef `_Compare` `key_compare`
- typedef `_Key` `key_type`
- typedef `_Tp` `mapped_type`
- typedef `_Base::pointer` `pointer`
- typedef `_Base::reference` `reference`
- typedef `std::reverse_iterator<iterator>` `reverse_iterator`
- typedef `_Base::size_type` `size_type`
- typedef `std::pair<const_Key,_Tp>` `value_type`

#### Public Member Functions

- **multimap** (`const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- `template<typename _InputIterator> multimap` (`_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- **multimap** (`const multimap &__x`)
- **multimap** (`const _Base &__x`)
- **multimap** (`multimap &&__x`) `noexcept(is_nothrow_copy_constructible<_Compare>::value)`
- **multimap** (`initializer_list<value_type> __l, const _Compare &__c=_Compare()`, `const allocator_type &__a=allocator_type()`)
- `void _M_attach` (`_Safe_iterator_base *__it, bool __constant`)
- `void _M_attach_single` (`_Safe_iterator_base *__it, bool __constant`) `throw ()`
- `_Base & _M_base` () `noexcept`
- `const _Base & _M_base` () `const noexcept`
- `void _M_detach` (`_Safe_iterator_base *__it`)
- `void _M_detach_single` (`_Safe_iterator_base *__it`) `throw ()`
- `void _M_invalidate_all` () `const`
- `void _M_invalidate_if` (`_Predicate __pred`)
- `void _M_transfer_from_if` (`_Safe_sequence &__from, _Predicate __pred`)
- **iterator begin** () `noexcept`
- **const\_iterator begin** () `const noexcept`
- **const\_iterator cbegin** () `const noexcept`
- **const\_iterator cend** () `const noexcept`
- `void clear` () `noexcept`
- **const\_reverse\_iterator crbegin** () `const noexcept`
- **const\_reverse\_iterator crend** () `const noexcept`
- `template<typename... _Args> iterator emplace` (`_Args &&... __args`)
- `template<typename... _Args> iterator emplace_hint` (`const_iterator __pos, _Args &&... __args`)
- **iterator end** () `noexcept`
- **const\_iterator end** () `const noexcept`
- `std::pair<iterator, iterator> equal_range` (`const key_type &__x`)
- `std::pair<const_iterator, const_iterator> equal_range` (`const key_type &__x`) `const`
- **iterator erase** (`const_iterator __position`)
- **iterator erase** (`iterator __position`)
- `size_type erase` (`const key_type &__x`)



- **iterator erase** ([const\\_iterator](#) \_\_first, [const\\_iterator](#) \_\_last)
- **iterator find** (const key\_type &\_\_x)
- **const\_iterator find** (const key\_type &\_\_x) const
- **iterator insert** (const [value\\_type](#) &\_\_x)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type> **iterator insert** (\_Pair &&\_\_x)
- void **insert** (std::initializer\_list< [value\\_type](#) > \_\_list)
- **iterator insert** ([const\\_iterator](#) \_\_position, const [value\\_type](#) &\_\_x)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type> **iterator insert** ([const\\_iterator](#) \_\_position, \_Pair &&\_\_x)
- template<typename \_InputIterator > void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- **iterator lower\_bound** (const key\_type &\_\_x)
- **const\_iterator lower\_bound** (const key\_type &\_\_x) const
- **multimap & operator=** (const [multimap](#) &\_\_x)
- **multimap & operator=** ([multimap](#) &&\_\_x)
- **multimap & operator=** (initializer\_list< [value\\_type](#) > \_\_l)
- **reverse\_iterator rbegin** () noexcept
- **const\_reverse\_iterator rbegin** () const noexcept
- **reverse\_iterator rend** () noexcept
- **const\_reverse\_iterator rend** () const noexcept
- void **swap** ([multimap](#) &\_\_x)
- **iterator upper\_bound** (const key\_type &\_\_x)
- **const\_iterator upper\_bound** (const key\_type &\_\_x) const

#### Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_const\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_iterators](#)
- unsigned int [\\_M\\_version](#)

#### Protected Member Functions

- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_revalidate\\_singular](#) ()
- void [\\_M\\_swap](#) ([\\_Safe\\_sequence\\_base](#) &\_\_x)

#### 4.411.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std::pair<const _Key, _Tp>>> class std::__debug::multimap<_Key, _Tp, _Compare, _Allocator>
```

Class std::multimap with safety/checking/debug instrumentation.

Definition at line 43 of file debug/multimap.h.

## 4.411.2 Member Function Documentation

4.411.2.1 void \_\_gnu\_debug::Safe\_sequence\_base::M\_attach ( \_Safe\_iterator\_base \* \_\_it, bool \_\_constant )  
[inherited]

Attach an iterator to this sequence.

4.411.2.2 void \_\_gnu\_debug::Safe\_sequence\_base::M\_attach\_single ( \_Safe\_iterator\_base \* \_\_it, bool \_\_constant ) throw  
[inherited]

Likewise but not thread safe.

4.411.2.3 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach ( \_Safe\_iterator\_base \* \_\_it ) [inherited]

Detach an iterator from this sequence

4.411.2.4 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_all ( ) [protected],[inherited]

Detach all iterators, leaving them singular.

Referenced by \_\_gnu\_debug::Safe\_sequence\_base::~~Safe\_sequence\_base().

4.411.2.5 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_single ( \_Safe\_iterator\_base \* \_\_it ) throw  
[inherited]

Likewise but not thread safe.

4.411.2.6 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_singular ( ) [protected],[inherited]

Detach all singular iterators.

## Postcondition

for all iterators i attached to this sequence, i->\_M\_version == \_M\_version.

4.411.2.7 \_\_gnu\_cxx::mutex& \_\_gnu\_debug::Safe\_sequence\_base::M\_get\_mutex ( ) throw [protected],  
[inherited]

For use in \_Safe\_sequence.

4.411.2.8 void \_\_gnu\_debug::Safe\_sequence\_base::M\_invalidate\_all ( ) const [inline],[inherited]

Invalidates all iterators.

Definition at line 233 of file safe\_base.h.

4.411.2.9 void \_\_gnu\_debug::Safe\_sequence< multimap<\_Key,\_Tp,\_Compare,\_Allocator> >::M\_invalidate\_if ( \_Predicate \_\_pred ) [inherited]

Invalidates all iterators x that reference this sequence, are not singular, and for which \_\_pred(x) returns true. \_\_pred will be invoked with the normal iterators nested in the safe ones.

4.411.2.10 void \_\_gnu\_debug::Safe\_sequence\_base::M\_revalidate\_singular ( ) [protected],[inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.411.2.11 `void __gnu_debug::Safe_sequence_base::M_swap ( _Safe_sequence_base & __x )` [protected],  
[inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.411.2.12 `void __gnu_debug::Safe_sequence< multimap< _Key, _Tp, _Compare, _Allocator > >::M_transfer_from_if`  
( `_Safe_sequence< multimap< _Key, _Tp, _Compare, _Allocator > > & __from, _Predicate __pred` )  
[inherited]

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

#### 4.411.3 Member Data Documentation

4.411.3.1 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_const_iterators` [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file `safe_base.h`.

4.411.3.2 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file `safe_base.h`.

4.411.3.3 `unsigned int __gnu_debug::Safe_sequence_base::M_version` [mutable],[inherited]

The container version number. This number may never be 0.

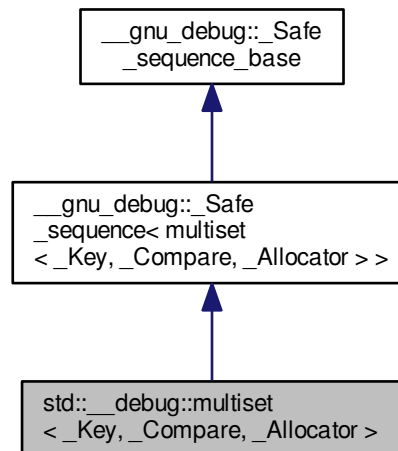
Definition at line 187 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [debug/multimap.h](#)

## 4.412 std::\_\_debug::multiset&lt; \_Key, \_Compare, \_Allocator &gt; Class Template Reference

Inheritance diagram for std::\_\_debug::multiset< \_Key, \_Compare, \_Allocator >:



## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::__Safe_iterator< _Base_const_iterator, multiset >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::__Safe_iterator< _Base_iterator, multiset >` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Compare` **value\_compare**
- typedef `_Key` **value\_type**

## Public Member Functions

- **multiset** (`const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- `template<typename _InputIterator > multiset` (`_InputIterator __first, _InputIterator __last, const _Compare &__comp=`  
`_Compare(), const _Allocator &__a=_Allocator()`)
- **multiset** (`const multiset &__x`)
- **multiset** (`const _Base &__x`)

- **multiset** ([multiset](#) &&\_\_x) noexcept(is\_nothrow\_copy\_constructible< [\\_Compare](#) >::value)
- **multiset** (initializer\_list< value\_type > \_\_l, const [\\_Compare](#) &\_\_comp=[\\_Compare](#)(), const allocator\_type &\_\_a=allocator\_type())
- void [\\_M\\_attach](#) ([\\_Safe\\_iterator\\_base](#) \*\_\_it, bool \_\_constant)
- void [\\_M\\_attach\\_single](#) ([\\_Safe\\_iterator\\_base](#) \*\_\_it, bool \_\_constant) throw ()
- [\\_Base](#) & [\\_M\\_base](#) () noexcept
- const [\\_Base](#) & [\\_M\\_base](#) () const noexcept
- void [\\_M\\_detach](#) ([\\_Safe\\_iterator\\_base](#) \*\_\_it)
- void [\\_M\\_detach\\_single](#) ([\\_Safe\\_iterator\\_base](#) \*\_\_it) throw ()
- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_invalidate\\_if](#) ([\\_Predicate](#) \_\_pred)
- void [\\_M\\_transfer\\_from\\_if](#) ([\\_Safe\\_sequence](#) &\_\_from, [\\_Predicate](#) \_\_pred)
- [iterator](#) **begin** () noexcept
- const [iterator](#) **begin** () const noexcept
- const [iterator](#) **cbegin** () const noexcept
- const [iterator](#) **cend** () const noexcept
- void **clear** () noexcept
- const [reverse\\_iterator](#) **crbegin** () const noexcept
- const [reverse\\_iterator](#) **crend** () const noexcept
- template<typename... [\\_Args](#)> [iterator](#) **emplace** ([\\_Args](#) &&...\_\_args)
- template<typename... [\\_Args](#)> [iterator](#) **emplace\_hint** (const [iterator](#) \_\_pos, [\\_Args](#) &&...\_\_args)
- [iterator](#) **end** () noexcept
- const [iterator](#) **end** () const noexcept
- std::pair< [iterator](#), [iterator](#) > **equal\_range** (const key\_type &\_\_x)
- std::pair< const [iterator](#), const [iterator](#) > **equal\_range** (const key\_type &\_\_x) const
- [iterator](#) **erase** (const [iterator](#) \_\_position)
- size\_type **erase** (const key\_type &\_\_x)
- [iterator](#) **erase** (const [iterator](#) \_\_first, const [iterator](#) \_\_last)
- [iterator](#) **find** (const key\_type &\_\_x)
- const [iterator](#) **find** (const key\_type &\_\_x) const
- [iterator](#) **insert** (const value\_type &\_\_x)
- [iterator](#) **insert** (value\_type &&\_\_x)
- [iterator](#) **insert** (const [iterator](#) \_\_position, const value\_type &\_\_x)
- [iterator](#) **insert** (const [iterator](#) \_\_position, value\_type &&\_\_x)
- template<typename [\\_InputIterator](#) > void **insert** ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last)
- void **insert** (initializer\_list< value\_type > \_\_l)
- [iterator](#) **lower\_bound** (const key\_type &\_\_x)
- const [iterator](#) **lower\_bound** (const key\_type &\_\_x) const
- [multiset](#) & **operator=** (const [multiset](#) &\_\_x)
- [multiset](#) & **operator=** ([multiset](#) &&\_\_x)
- [multiset](#) & **operator=** (initializer\_list< value\_type > \_\_l)
- [reverse\\_iterator](#) **rbegin** () noexcept
- const [reverse\\_iterator](#) **rbegin** () const noexcept
- [reverse\\_iterator](#) **rend** () noexcept
- const [reverse\\_iterator](#) **rend** () const noexcept
- void **swap** ([multiset](#) &\_\_x)
- [iterator](#) **upper\_bound** (const key\_type &\_\_x)
- const [iterator](#) **upper\_bound** (const key\_type &\_\_x) const

## Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_const\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_iterators](#)
- unsigned int [\\_M\\_version](#)

## Protected Member Functions

- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_revalidate\\_singular](#) ()
- void [\\_M\\_swap](#) ([\\_Safe\\_sequence\\_base](#) &\_\_x)

## 4.412.1 Detailed Description

template<typename \_Key, typename \_Compare = std::less<\_Key>, typename \_Allocator = std::allocator<\_Key>>class std::\_\_debug::multiset<\_Key, \_Compare, \_Allocator>

Class std::multiset with safety/checking/debug instrumentation.

Definition at line 43 of file debug/multiset.h.

## 4.412.2 Member Function Documentation

4.412.2.1 void [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::M\\_attach](#) ( [\\_Safe\\_iterator\\_base](#) \* \_\_it, bool \_\_constant )  
[inherited]

Attach an iterator to this sequence.

4.412.2.2 void [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::M\\_attach\\_single](#) ( [\\_Safe\\_iterator\\_base](#) \* \_\_it, bool \_\_constant ) throw()  
[inherited]

Likewise but not thread safe.

4.412.2.3 void [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::M\\_detach](#) ( [\\_Safe\\_iterator\\_base](#) \* \_\_it ) [inherited]

Detach an iterator from this sequence

4.412.2.4 void [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::M\\_detach\\_all](#) ( ) [protected], [inherited]

Detach all iterators, leaving them singular.

Referenced by [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::~~Safe\\_sequence\\_base\(\)](#).

4.412.2.5 void [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::M\\_detach\\_single](#) ( [\\_Safe\\_iterator\\_base](#) \* \_\_it ) throw()  
[inherited]

Likewise but not thread safe.

4.412.2.6 void [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::M\\_detach\\_singular](#) ( ) [protected], [inherited]

Detach all singular iterators.

**Postcondition**

for all iterators *i* attached to this sequence, *i*->\_M\_version == \_M\_version.

**4.412.2.7** `__gnu_cxx::__mutex& __gnu_debug::Safe_sequence_base::M_get_mutex ( ) throw` `[protected]`,  
`[inherited]`

For use in `_Safe_sequence`.

**4.412.2.8** `void __gnu_debug::Safe_sequence_base::M_invalidate_all ( ) const` `[inline]`, `[inherited]`

Invalidates all iterators.

Definition at line 233 of file `safe_base.h`.

**4.412.2.9** `void __gnu_debug::Safe_sequence< multiset< _Key, _Compare, _Allocator > >::M_invalidate_if ( _Predicate __pred )` `[inherited]`

Invalidates all iterators *x* that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

**4.412.2.10** `void __gnu_debug::Safe_sequence_base::M_revalidate_singular ( )` `[protected]`, `[inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

**4.412.2.11** `void __gnu_debug::Safe_sequence_base::M_swap ( _Safe_sequence_base & __x )` `[protected]`,  
`[inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**4.412.2.12** `void __gnu_debug::Safe_sequence< multiset< _Key, _Compare, _Allocator > >::M_transfer_from_if ( _Safe_sequence< multiset< _Key, _Compare, _Allocator > > & __from, _Predicate __pred )` `[inherited]`

Transfers all iterators *x* that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

**4.412.3 Member Data Documentation**

**4.412.3.1** `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_const_iterators` `[inherited]`

The list of constant iterators that reference this container.

Definition at line 184 of file `safe_base.h`.

**4.412.3.2** `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_iterators` `[inherited]`

The list of mutable iterators that reference this container.

Definition at line 181 of file `safe_base.h`.

**4.412.3.3** `unsigned int __gnu_debug::Safe_sequence_base::M_version` `[mutable]`, `[inherited]`

The container version number. This number may never be 0.

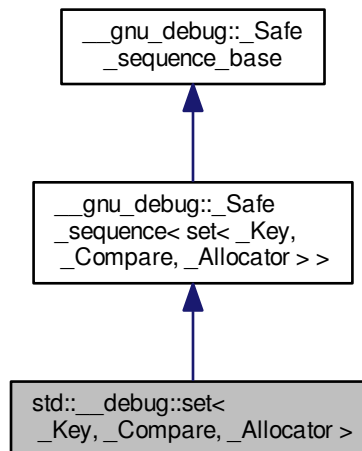
Definition at line 187 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [debug/multiset.h](#)

4.413 `std::__debug::set<_Key,_Compare,_Allocator>` Class Template Reference

Inheritance diagram for `std::__debug::set<_Key,_Compare,_Allocator>`:



## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::Safe_iterator<_Base_const_iterator, set>` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator<const_iterator>` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::Safe_iterator<_Base_iterator, set>` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator<iterator>` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Compare` **value\_compare**
- typedef `_Key` **value\_type**

## Public Member Functions

- **set** (`const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)



- `template<typename _InputIterator > set (_InputIterator __first, _InputIterator __last, const _Compare &__comp=↵  
Compare(), const _Allocator &__a=_Allocator())`
- `set (const set &__x)`
- `set (const _Base &__x)`
- `set (set &&__x) noexcept(is_nothrow_copy_constructible< _Compare >::value)`
- `set (initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__↵  
a=allocator_type())`
- `void _M_attach (_Safe_iterator_base *__it, bool __constant)`
- `void _M_attach_single (_Safe_iterator_base *__it, bool __constant) throw ()`
- `_Base & _M_base () noexcept`
- `const _Base & _M_base () const noexcept`
- `void _M_detach (_Safe_iterator_base *__it)`
- `void _M_detach_single (_Safe_iterator_base *__it) throw ()`
- `void _M_invalidate_all () const`
- `void _M_invalidate_if (_Predicate __pred)`
- `void _M_transfer_from_if (_Safe_sequence &__from, _Predicate __pred)`
- `iterator begin () noexcept`
- `const_iterator begin () const noexcept`
- `const_iterator cbegin () const noexcept`
- `const_iterator cend () const noexcept`
- `void clear () noexcept`
- `const_reverse_iterator crbegin () const noexcept`
- `const_reverse_iterator crend () const noexcept`
- `template<typename... _Args> std::pair< iterator, bool > emplace (_Args &&...__args)`
- `template<typename... _Args> iterator emplace_hint (const_iterator __pos, _Args &&...__args)`
- `iterator end () noexcept`
- `const_iterator end () const noexcept`
- `std::pair< iterator, iterator > equal_range (const key_type &__x)`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const`
- `iterator erase (const_iterator __position)`
- `size_type erase (const key_type &__x)`
- `iterator erase (const_iterator __first, const_iterator __last)`
- `iterator find (const key_type &__x)`
- `const_iterator find (const key_type &__x) const`
- `std::pair< iterator, bool > insert (const value_type &__x)`
- `std::pair< iterator, bool > insert (value_type &&__x)`
- `iterator insert (const_iterator __position, const value_type &__x)`
- `iterator insert (const_iterator __position, value_type &&__x)`
- `template<typename _InputIterator > void insert (_InputIterator __first, _InputIterator __last)`
- `void insert (initializer_list< value_type > __l)`
- `iterator lower_bound (const key_type &__x)`
- `const_iterator lower_bound (const key_type &__x) const`
- `set & operator= (const set &__x)`
- `set & operator= (set &&__x)`
- `set & operator= (initializer_list< value_type > __l)`
- `reverse_iterator rbegin () noexcept`
- `const_reverse_iterator rbegin () const noexcept`
- `reverse_iterator rend () noexcept`
- `const_reverse_iterator rend () const noexcept`
- `void swap (set &__x)`
- `iterator upper_bound (const key_type &__x)`
- `const_iterator upper_bound (const key_type &__x) const`

## Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_const\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_iterators](#)
- unsigned int [\\_M\\_version](#)

## Protected Member Functions

- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_revalidate\\_singular](#) ()
- void [\\_M\\_swap](#) ([\\_Safe\\_sequence\\_base](#) &\_\_x)

## 4.413.1 Detailed Description

template<typename [\\_Key](#), typename [\\_Compare](#) = [std::less<\\_Key>](#), typename [\\_Allocator](#) = [std::allocator<\\_Key>](#)>>class [std::\\_\\_debug::set<\\_Key, \\_Compare, \\_Allocator>](#)

Class [std::set](#) with safety/checking/debug instrumentation.

Definition at line 43 of file [debug/set.h](#).

## 4.413.2 Member Function Documentation

4.413.2.1 void [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::M\\_attach](#) ( [\\_Safe\\_iterator\\_base](#) \* \_\_it, bool \_\_constant )  
[[inherited](#)]

Attach an iterator to this sequence.

4.413.2.2 void [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::M\\_attach\\_single](#) ( [\\_Safe\\_iterator\\_base](#) \* \_\_it, bool \_\_constant ) throw()  
[[inherited](#)]

Likewise but not thread safe.

4.413.2.3 void [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::M\\_detach](#) ( [\\_Safe\\_iterator\\_base](#) \* \_\_it ) [[inherited](#)]

Detach an iterator from this sequence

4.413.2.4 void [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::M\\_detach\\_all](#) ( ) [[protected](#)], [[inherited](#)]

Detach all iterators, leaving them singular.

Referenced by [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::~~Safe\\_sequence\\_base\(\)](#).

4.413.2.5 void [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::M\\_detach\\_single](#) ( [\\_Safe\\_iterator\\_base](#) \* \_\_it ) throw()  
[[inherited](#)]

Likewise but not thread safe.

4.413.2.6 void [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::M\\_detach\\_singular](#) ( ) [[protected](#)], [[inherited](#)]

Detach all singular iterators.

**Postcondition**

for all iterators *i* attached to this sequence, *i*->\_M\_version == \_M\_version.

**4.413.2.7** `__gnu_cxx::__mutex& __gnu_debug::Safe_sequence_base::M_get_mutex ( ) throw` `[protected]`,  
`[inherited]`

For use in `_Safe_sequence`.

**4.413.2.8** `void __gnu_debug::Safe_sequence_base::M_invalidate_all ( ) const` `[inline]`, `[inherited]`

Invalidates all iterators.

Definition at line 233 of file `safe_base.h`.

**4.413.2.9** `void __gnu_debug::Safe_sequence< set< _Key, _Compare, _Allocator > >::M_invalidate_if ( _Predicate  
__pred )` `[inherited]`

Invalidates all iterators *x* that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

**4.413.2.10** `void __gnu_debug::Safe_sequence_base::M_revalidate_singular ( )` `[protected]`, `[inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

**4.413.2.11** `void __gnu_debug::Safe_sequence_base::M_swap ( _Safe_sequence_base & __x )` `[protected]`,  
`[inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**4.413.2.12** `void __gnu_debug::Safe_sequence< set< _Key, _Compare, _Allocator > >::M_transfer_from_if (`  
`_Safe_sequence< set< _Key, _Compare, _Allocator > > & __from, _Predicate __pred )` `[inherited]`

Transfers all iterators *x* that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

**4.413.3 Member Data Documentation**

**4.413.3.1** `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_const_iterators` `[inherited]`

The list of constant iterators that reference this container.

Definition at line 184 of file `safe_base.h`.

**4.413.3.2** `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_iterators` `[inherited]`

The list of mutable iterators that reference this container.

Definition at line 181 of file `safe_base.h`.

**4.413.3.3** `unsigned int __gnu_debug::Safe_sequence_base::M_version` `[mutable]`, `[inherited]`

The container version number. This number may never be 0.

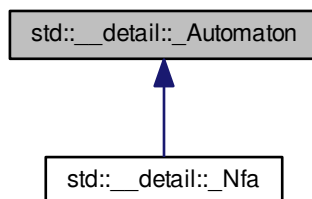
Definition at line 187 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [debug/set.h](#)

#### 4.414 std::\_\_detail::\_Automaton Class Reference

Inheritance diagram for std::\_\_detail::\_Automaton:



##### Public Types

- typedef unsigned int **\_SizeT**

##### Public Member Functions

- virtual \_SizeT **\_M\_sub\_count** () const =0

##### 4.414.1 Detailed Description

Base class for, um, automata. Could be an NFA or a DFA. Your choice.

Definition at line 43 of file regex\_nfa.h.

The documentation for this class was generated from the following file:

- [regex\\_nfa.h](#)

#### 4.415 std::\_\_detail::\_Before\_begin< \_NodeAlloc > Struct Template Reference

Inherits `_NodeAlloc`.

##### Public Member Functions

- **\_Before\_begin** (const [\\_Before\\_begin](#) &)=default
- **\_Before\_begin** ([\\_Before\\_begin](#) &&)=default
- template<typename `_Alloc` > **\_Before\_begin** (`_Alloc` &&`_a`)

## Public Attributes

- [\\_Hash\\_node\\_base](#) **\_M\_node**

## 4.415.1 Detailed Description

```
template<typename _NodeAlloc>struct std::__detail::_Before_begin< _NodeAlloc >
```

This type is to combine a `_Hash_node_base` instance with an allocator instance through inheritance to benefit from EBO when possible.

Definition at line 1652 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

4.416 `std::__detail::_CharMatcher< _InIterT, _TraitsT >` Struct Template Reference

## Public Types

- typedef `_TraitsT::char_type` **char\_type**

## Public Member Functions

- **\_CharMatcher** (`char_type __c`, `const _TraitsT &__t=_TraitsT()`)
- **operator()** (`const \_PatternCursor &__pc`) `const`

## Public Attributes

- `char_type` **\_M\_c**
- `const _TraitsT &` **\_M\_traits**

## 4.416.1 Detailed Description

```
template<typename _InIterT, typename _TraitsT>struct std::__detail::_CharMatcher< _InIterT, _TraitsT >
```

Matches a single character.

Definition at line 129 of file `regex_nfa.h`.

The documentation for this struct was generated from the following file:

- [regex\\_nfa.h](#)

4.417 `std::__detail::_Compiler< _InIter, _TraitsT >` Class Template Reference

## Public Types

- typedef `std::iterator_traits< _InIter >::value_type` **\_CharT**
- typedef [regex\\_constants::syntax\\_option\\_type](#) **\_FlagT**
- typedef `_InIter` **\_IterT**
- typedef [std::basic\\_string](#)< `_CharT` > **\_StringT**

## Public Member Functions

- **\_Compiler** (const \_InIter &\_\_b, const \_InIter &\_\_e, \_TraitsT &\_\_traits, \_FlagT \_\_flags)
- const [\\_Nfa](#) & **\_M\_nfa** () const

## 4.417.1 Detailed Description

```
template<typename _InIter, typename _TraitsT>class std::__detail::__Compiler< _InIter, _TraitsT >
```

Builds an NFA from an input iterator interval.

Definition at line 634 of file `regex_compiler.h`.

The documentation for this class was generated from the following file:

- [regex\\_compiler.h](#)

## 4.418 std::\_\_detail::\_\_Default\_ranged\_hash Struct Reference

## 4.418.1 Detailed Description

Default ranged hash function H. In principle it should be a function object composed from objects of type H1 and H2 such that  $h(k, N) = h2(h1(k), N)$ , but that would mean making extra copies of h1 and h2. So instead we'll just use a tag to tell class template hashtable to do that composition.

Definition at line 353 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.419 std::\_\_detail::\_\_EndTagger&lt; \_FwdIterT, \_TraitsT &gt; Struct Template Reference

## Public Member Functions

- **\_EndTagger** (int \_\_i)
- void **operator()** (const [\\_PatternCursor](#) &\_\_pc, [\\_Results](#) &\_\_r)

## Public Attributes

- int **\_M\_index**
- [\\_FwdIterT](#) **\_M\_pos**

## 4.419.1 Detailed Description

```
template<typename _FwdIterT, typename _TraitsT>struct std::__detail::__EndTagger< _FwdIterT, _TraitsT >
```

End state tag.

Definition at line 104 of file `regex_nfa.h`.

The documentation for this struct was generated from the following file:

- [regex\\_nfa.h](#)

#### 4.420 `std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, __cache_hash_code >` Struct Template Reference

##### 4.420.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _Equal, typename _HashCodeType, bool __cache_hash_code>struct std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, __cache_hash_code >
```

Primary class template `_Equal_helper`.

Definition at line 1156 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.421 `std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, false >` Struct Template Reference

##### Static Public Member Functions

- static bool `_S_equals` (const `_Equal` &\_\_eq, const `_ExtractKey` &\_\_extract, const `_Key` &\_\_k, `_HashCodeType`, `_Hash_node`< `_Value`, false > \*\_\_n)

##### 4.421.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _Equal, typename _HashCodeType>struct std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, false >
```

Specialization.

Definition at line 1172 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.422 `std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, true >` Struct Template Reference

##### Static Public Member Functions

- static bool `_S_equals` (const `_Equal` &\_\_eq, const `_ExtractKey` &\_\_extract, const `_Key` &\_\_k, `_HashCodeType` \_\_c, `_Hash_node`< `_Value`, true > \*\_\_n)

##### 4.422.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _Equal, typename _HashCodeType>struct std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, true >
```

Specialization.

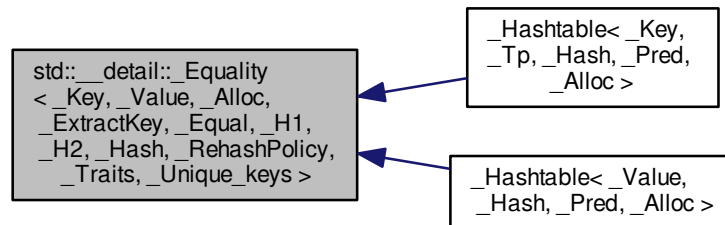
Definition at line 1161 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.423 `std::__detail::__Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >` Struct Template Reference

Inheritance diagram for `std::__detail::__Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >`:



##### 4.423.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename
_H2, typename _Hash, typename _RehashPolicy, typename _Traits, bool _Unique_keys = _Traits::__unique_keys::value>struct std::__
__detail::__Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >
```

Primary class template `_Equality`.

This is for implementing equality comparison for unordered containers, per N3068, by John Lakos and Pablo Halpern. Algorithmically, we follow closely the reference implementations therein.

Definition at line 1559 of file `hashtable_policy.h`.

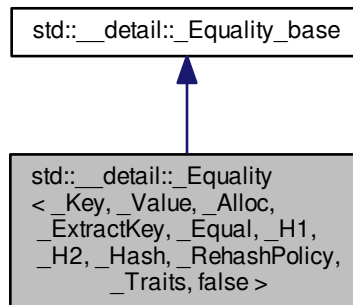
The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)



#### 4.424 `std::__detail::Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >` Struct Template Reference

Inheritance diagram for `std::__detail::Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >`:



##### Public Types

- using `__hashtable` = `__Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`

##### Public Member Functions

- `bool __M_equal (const __hashtable &) const`

##### Static Protected Member Functions

- `template<typename _Uiterator > static bool __S_is_permutation (_Uiterator, _Uiterator, _Uiterator)`

##### 4.424.1 Detailed Description

`template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2, typename _Hash, typename _RehashPolicy, typename _Traits> struct std::__detail::Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >`

Specialization.

Definition at line 1604 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.425 `std::__detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >` Struct Template Reference

##### Public Types

- using `__hashtable` = `_Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`

##### Public Member Functions

- `bool _M_equal (const __hashtable &) const`

##### 4.425.1 Detailed Description

template<typename \_Key, typename \_Value, typename \_Alloc, typename \_ExtractKey, typename \_Equal, typename \_H1, typename \_H2, typename \_Hash, typename \_RehashPolicy, typename \_Traits> struct `std::__detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >`

Specialization.

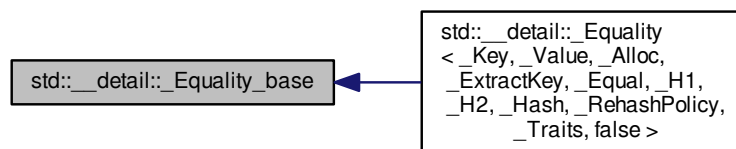
Definition at line 1566 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.426 `std::__detail::_Equality_base` Struct Reference

Inheritance diagram for `std::__detail::_Equality_base`:



##### Static Protected Member Functions

- template<typename \_Uiterator > static `bool _S_is_permutation (_Uiterator, _Uiterator, _Uiterator)`

##### 4.426.1 Detailed Description

struct `_Equality_base`.

Common types and functions for class `_Equality`.

Definition at line 1492 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.427 std::\_\_detail::\_Grep\_matcher Class Reference

##### Public Member Functions

- **\_Grep\_matcher** ([\\_PatternCursor](#) &\_\_p, [\\_Results](#) &\_\_r, const [\\_AutomatonPtr](#) &\_\_automaton, [regex\\_constants::match\\_flag\\_type](#) \_\_flags)

##### 4.427.1 Detailed Description

Executes a regular expression NFA/DFA over a range using a variant of the parallel execution algorithm featured in the grep utility, modified to use Laurikari tags.

Definition at line 110 of file regex\_grep\_matcher.h.

The documentation for this class was generated from the following file:

- [regex\\_grep\\_matcher.h](#)

#### 4.428 std::\_\_detail::\_Hash\_code\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_cache\_hash\_code > Struct Template Reference

##### 4.428.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache_hash_code> struct std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >
```

Primary class template `_Hash_code_base`.

Encapsulates two policy issues that aren't quite orthogonal. (1) the difference between using a ranged hash function and using the combination of a hash function and a range-hashing function. In the former case we don't have such things as hash codes, so we have a dummy type as placeholder. (2) Whether or not we cache hash codes. Caching hash codes is meaningless if we have a ranged hash function.

We also put the key extraction objects here, for convenience. Each specialization derives from one or more of the template parameters to benefit from Ebo. This is important as this type is inherited in some cases by the `_Local_iterator_base` type used to implement `local_iterator` and `const_local_iterator`. As with any iterator type we prefer to make it as small as possible.

Primary template is unused except as a hook for specializations.

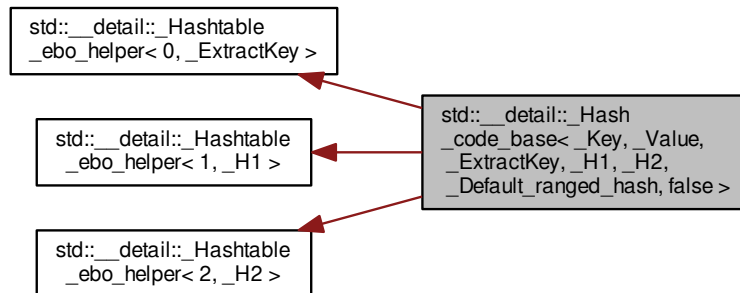
Definition at line 907 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.429 `std::__detail::_Hash_code_base<_Key,_Value,_ExtractKey,_H1,_H2,_Default_ranged_hash,false>` Struct Template Reference

Inheritance diagram for `std::__detail::_Hash_code_base<_Key,_Value,_ExtractKey,_H1,_H2,_Default_ranged_hash,false>`:



#### Public Types

- typedef `_H1 hasher`

#### Public Member Functions

- hasher **hash\_function** () const

#### Protected Types

- typedef `std::size_t __hash_code`
- typedef `_Hash_node<_Value,false> __node_type`

#### Protected Member Functions

- **\_Hash\_code\_base** (const `_ExtractKey` &\_\_ex, const `_H1` &\_\_h1, const `_H2` &\_\_h2, const `_Default_ranged_hash` &\_\_)
- `std::size_t _M_bucket_index` (const `_Key` &\_\_, \_\_hash\_code \_\_c, `std::size_t` \_\_n) const
- `std::size_t _M_bucket_index` (const `__node_type` \* \_\_p, `std::size_t` \_\_n) const
- void **\_M\_copy\_code** (`__node_type` \*, const `__node_type` \*) const
- const `_ExtractKey` & **\_M\_extract** () const
- `_ExtractKey` & **\_M\_extract** ()
- const `_H1` & **\_M\_h1** () const
- `_H1` & **\_M\_h1** ()
- const `_H2` & **\_M\_h2** () const
- `_H2` & **\_M\_h2** ()
- \_\_hash\_code **\_M\_hash\_code** (const `_Key` &\_\_k) const
- void **\_M\_store\_code** (`__node_type` \*, \_\_hash\_code) const
- void **\_M\_swap** (`_Hash_code_base` &\_\_x)

#### 4.429.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2> struct std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false >
```

Specialization: hash function and range-hashing function, no caching of hash codes. Provides typedef and accessor required by C++ 11.

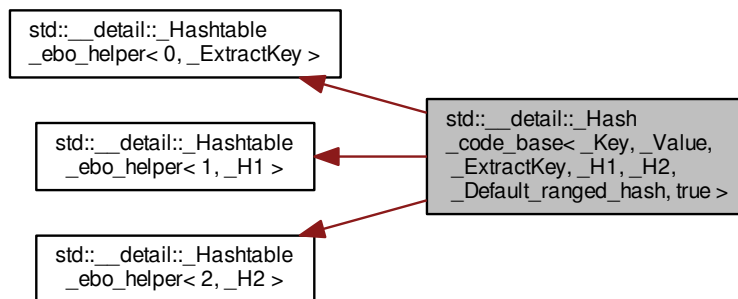
Definition at line 987 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.430 std::\_\_detail::\_Hash\_code\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Default\_ranged\_hash, true > Struct Template Reference

Inheritance diagram for std::\_\_detail::\_Hash\_code\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Default\_ranged\_hash, true >:



#### Public Types

- typedef `_H1` **hasher**

#### Public Member Functions

- hasher **hash\_function** () const

#### Protected Types

- typedef `std::size_t` **\_\_hash\_code**
- typedef `_Hash_node< _Value, true >` **\_\_node\_type**

#### Protected Member Functions

- `_Hash_code_base` (const `_ExtractKey` &\_\_ex, const `_H1` &\_\_h1, const `_H2` &\_\_h2, const `_Default_ranged_hash` &)
- `std::size_t _M_bucket_index` (const `_Key` &, `__hash_code` \_\_c, `std::size_t` \_\_n) const
- `std::size_t _M_bucket_index` (const `__node_type` \*\_\_p, `std::size_t` \_\_n) const
- `void _M_copy_code` (`__node_type` \*\_\_to, const `__node_type` \*\_\_from) const
- const `_ExtractKey` & `_M_extract` () const
- `_ExtractKey` & `_M_extract` ()
- const `_H1` & `_M_h1` () const
- `_H1` & `_M_h1` ()
- const `_H2` & `_M_h2` () const
- `_H2` & `_M_h2` ()
- `__hash_code` `_M_hash_code` (const `_Key` &\_\_k) const
- `void _M_store_code` (`__node_type` \*\_\_n, `__hash_code` \_\_c) const
- `void _M_swap` (`_Hash_code_base` &\_\_x)

#### Friends

- struct `_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true >`

#### 4.430.1 Detailed Description

template<typename `_Key`, typename `_Value`, typename `_ExtractKey`, typename `_H1`, typename `_H2`>struct `std::__detail::__Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true >`

Specialization: hash function and range-hashing function, caching hash codes. H is provided but ignored. Provides typedef and accessor required by C++ 11.

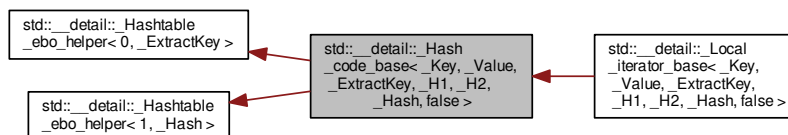
Definition at line 1070 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.431 `std::__detail::__Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >` Struct Template Reference

Inheritance diagram for `std::__detail::__Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >`:



## Protected Types

- typedef void \* **\_\_hash\_code**
- typedef [\\_Hash\\_node](#)< \_Value, false > **\_\_node\_type**

## Protected Member Functions

- **\_Hash\_code\_base** (const \_ExtractKey &\_\_ex, const \_H1 &, const \_H2 &, const \_Hash &\_\_h)
- std::size\_t **\_M\_bucket\_index** (const \_Key &\_\_k, \_\_hash\_code, std::size\_t \_\_n) const
- std::size\_t **\_M\_bucket\_index** (const [\\_\\_node\\_type](#) \*\_\_p, std::size\_t \_\_n) const
- void **\_M\_copy\_code** ([\\_\\_node\\_type](#) \*, const [\\_\\_node\\_type](#) \*) const
- const \_ExtractKey & **\_M\_extract** () const
- \_ExtractKey & **\_M\_extract** ()
- \_\_hash\_code **\_M\_hash\_code** (const \_Key &\_\_key) const
- const \_Hash & **\_M\_ranged\_hash** () const
- \_Hash & **\_M\_ranged\_hash** ()
- void **\_M\_store\_code** ([\\_\\_node\\_type](#) \*, \_\_hash\_code) const
- void **\_M\_swap** ([\\_Hash\\_code\\_base](#) &\_\_x)

### 4.431.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash> struct std::__detail::__Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >
```

Specialization: ranged hash function, no caching hash codes. H1 and H2 are provided but ignored. We define a dummy hash code type.

Definition at line 913 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.432 std::\_\_detail::\_\_Hash\_node< \_Value, \_Cache\_hash\_code > Struct Template Reference

### 4.432.1 Detailed Description

```
template<typename _Value, bool _Cache_hash_code> struct std::__detail::__Hash_node< _Value, _Cache_hash_code >
```

Primary template struct `_Hash_node`.

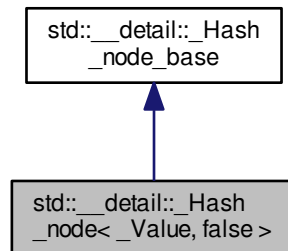
Definition at line 162 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.433 std::\_\_detail::\_Hash\_node&lt; \_Value, false &gt; Struct Template Reference

Inheritance diagram for std::\_\_detail::\_Hash\_node< \_Value, false >:



## Public Member Functions

- `template<typename... _Args> _Hash_node ( _Args &&...__args)`
- `\_Hash\_node * _M_next () const`

## Public Attributes

- `\_Hash\_node\_base * _M_nxt`
- `_Value _M_v`

## 4.433.1 Detailed Description

`template<typename _Value>struct std::__detail::_Hash_node< _Value, false >`

Specialization for nodes without caches, struct `_Hash_node`.

Base class is `__detail::_Hash_node_base`.

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

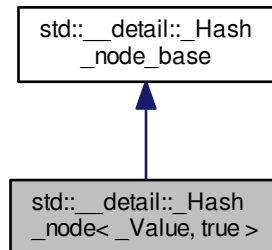
The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)



#### 4.434 `std::__detail::_Hash_node<_Value, true>` Struct Template Reference

Inheritance diagram for `std::__detail::_Hash_node<_Value, true>`:



##### Public Member Functions

- `template<typename... _Args> _Hash_node (_Args &&...__args)`
- `\_Hash\_node * _M_next () const`

##### Public Attributes

- `std::size_t _M_hash_code`
- `\_Hash\_node\_base * _M_nxt`
- `_Value _M_v`

##### 4.434.1 Detailed Description

`template<typename _Value> struct std::__detail::_Hash_node<_Value, true>`

Specialization for nodes with caches, struct `_Hash_node`.

Base class is `__detail::_Hash_node_base`.

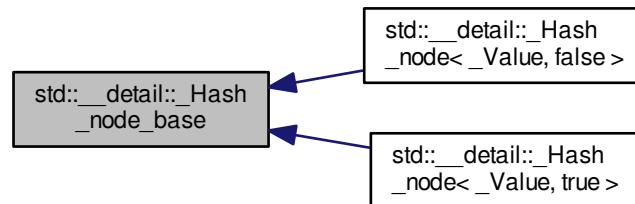
Definition at line 170 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.435 std::\_\_detail::\_Hash\_node\_base Struct Reference

Inheritance diagram for std::\_\_detail::\_Hash\_node\_base:



## Public Member Functions

- `_Hash_node_base` ([\\_Hash\\_node\\_base](#) \* \_\_next)

## Public Attributes

- [\\_Hash\\_node\\_base](#) \* `_M_nxt`

## 4.435.1 Detailed Description

struct `_Hash_node_base`

Nodes, used to wrap elements stored in the hash table. A policy template parameter of class template `_Hashtable` controls whether nodes also store a hash code. In some cases (e.g. strings) this may be a performance win.

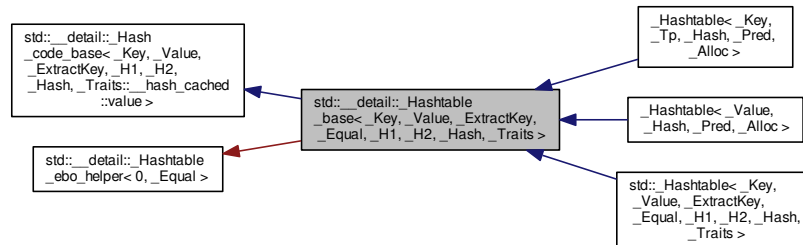
Definition at line 149 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.436 `std::__detail::_Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >` Struct Template Reference

Inheritance diagram for `std::__detail::_Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >`:



#### Public Types

- using **\_\_constant\_iterators** = typename \_\_traits\_type::\_\_constant\_iterators
- using **\_\_hash\_cached** = typename \_\_traits\_type::\_\_hash\_cached
- using **\_\_hash\_code** = typename \_\_hash\_code\_base::\_\_hash\_code
- using **\_\_hash\_code\_base** = [\\_Hash\\_code\\_base](#)< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_hash\_cached::value >
- using **\_\_iconv\_type** = typename std::conditional< \_\_unique\_keys::value, \_Select1st, \_Identity >::type
- using **\_\_ireturn\_type** = typename std::conditional< \_\_unique\_keys::value, [std::pair](#)< [iterator](#), bool >, [iterator](#) >::type
- using **\_\_node\_type** = typename \_\_hash\_code\_base::\_\_node\_type
- using **\_\_traits\_type** = \_Traits
- using **\_\_unique\_keys** = typename \_\_traits\_type::\_\_unique\_keys
- using **const\_iterator** = [\\_\\_detail::\\_Node\\_const\\_iterator](#)< value\_type, \_\_constant\_iterators::value, \_\_hash\_cached::value >
- using **const\_local\_iterator** = [\\_\\_detail::\\_Local\\_const\\_iterator](#)< key\_type, value\_type, \_ExtractKey, \_H1, \_H2, \_\_Hash, \_\_constant\_iterators::value, \_\_hash\_cached::value >
- typedef std::ptrdiff\_t **difference\_type**
- using **iterator** = [\\_\\_detail::\\_Node\\_iterator](#)< value\_type, \_\_constant\_iterators::value, \_\_hash\_cached::value >
- typedef \_Equal **key\_equal**
- typedef \_Key **key\_type**
- using **local\_iterator** = [\\_\\_detail::\\_Local\\_iterator](#)< key\_type, value\_type, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_constant\_iterators::value, \_\_hash\_cached::value >
- typedef std::size\_t **size\_type**
- typedef \_Value **value\_type**

#### Protected Types

- using **\_\_bucket\_type** = [\\_\\_node\\_base](#) \*
- using **\_\_node\_base** = [\\_\\_detail::\\_Hash\\_node\\_base](#)

## Protected Member Functions

- **\_Hashtable\_base** (const \_ExtractKey &\_\_ex, const \_H1 &\_\_h1, const \_H2 &\_\_h2, const \_Hash &\_\_hash, const \_Equal &\_\_eq)
- const \_Equal & **\_M\_eq** () const
- \_Equal & **\_M\_eq** ()
- bool **\_M\_equals** (const \_Key &\_\_k, \_\_hash\_code \_\_c, \_\_node\_type \*\_\_n) const
- void **\_M\_swap** (\_Hashtable\_base &\_\_x)

## 4.436.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _Equal, typename _H1, typename _H2, typename _Hash,
typename _Traits> struct std::__detail::_Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >
```

Primary class template \_Hashtable\_base.

Helper class adding management of \_Equal functor to \_Hash\_code\_base type.

Base class templates are:

- \_\_detail::\_Hash\_code\_base
- \_\_detail::\_Hashtable\_ebo\_helper

Definition at line 58 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.437 std::\_\_detail::\_Hashtable\_ebo\_helper&lt; \_Nm, \_Tp, \_\_use\_ebo &gt; Struct Template Reference

## 4.437.1 Detailed Description

```
template<int _Nm, typename _Tp, bool __use_ebo = !_is_final(_Tp) && __is_empty(_Tp)> struct std::__detail::_Hashtable_ebo_↵
helper< _Nm, _Tp, __use_ebo >
```

Primary class template \_Hashtable\_ebo\_helper.

Helper class using EBO when it is not forbidden, type is not final, and when it worth it, type is empty.

Definition at line 831 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.438 std::\_\_detail::\_Hashtable\_ebo\_helper&lt; \_Nm, \_Tp, false &gt; Struct Template Reference

## Public Member Functions

- **\_Hashtable\_ebo\_helper** (const \_Tp &\_\_tp)

### Static Public Member Functions

- static const `_Tp & _S_cget` (const [\\_Hashtable\\_ebo\\_helper](#) &\_\_eboh)
- static `_Tp & _S_get` ([\\_Hashtable\\_ebo\\_helper](#) &\_\_eboh)

#### 4.438.1 Detailed Description

```
template<int _Nm, typename _Tp>struct std::__detail::__Hashtable_ebo_helper< _Nm, _Tp, false >
```

Specialization not using EBO.

Definition at line 854 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

### 4.439 `std::__detail::__Hashtable_ebo_helper< _Nm, _Tp, true >` Struct Template Reference

Inherits `_Tp`.

### Public Member Functions

- `_Hashtable_ebo_helper` (const `_Tp` &\_\_tp)

### Static Public Member Functions

- static const `_Tp & _S_cget` (const [\\_Hashtable\\_ebo\\_helper](#) &\_\_eboh)
- static `_Tp & _S_get` ([\\_Hashtable\\_ebo\\_helper](#) &\_\_eboh)

#### 4.439.1 Detailed Description

```
template<int _Nm, typename _Tp>struct std::__detail::__Hashtable_ebo_helper< _Nm, _Tp, true >
```

Specialization using EBO.

Definition at line 835 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

### 4.440 `std::__detail::__Hashtable_traits< _Cache_hash_code, _Constant_iterators, _Unique_keys >` Struct Template Reference

### Public Types

- `template<bool _Cond> using __bool_constant = integral_constant< bool, _Cond >`
- `using __constant_iterators = __bool_constant< _Constant_iterators >`
- `using __hash_cached = __bool_constant< _Cache_hash_code >`
- `using __unique_keys = __bool_constant< _Unique_keys >`

### 4.440.1 Detailed Description

```
template<bool _Cache_hash_code, bool _Constant_iterators, bool _Unique_keys>struct std::__detail::_Hashtable_traits<_Cache↵
_hash_code, _Constant_iterators, _Unique_keys >
```

struct \_Hashtable\_traits

Important traits for hash tables.

#### Template Parameters

<code>_Cache_hash_code</code>	Boolean value. True if the value of the hash function is stored along with the value. This is a time-space tradeoff. Storing it may improve lookup speed by reducing the number of times we need to call the <code>_Equal</code> function.
<code>_Constant_iterators</code>	Boolean value. True if iterator and const_iterator are both constant iterator types. This is true for <code>unordered_set</code> and <code>unordered_multiset</code> , false for <code>unordered_map</code> and <code>unordered_multimap</code> .
<code>_Unique_keys</code>	Boolean value. True if the return value of <code>_Hashtable::count(k)</code> is always at most one, false if it may be an arbitrary number. This is true for <code>unordered_set</code> and <code>unordered↵_map</code> , false for <code>unordered_multiset</code> and <code>unordered_multimap</code> .

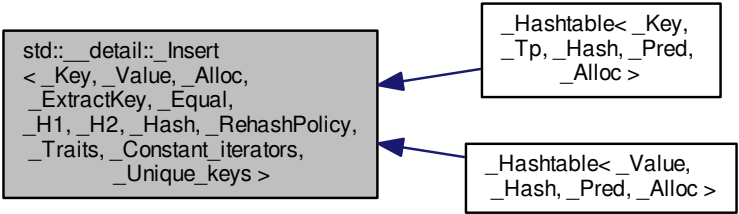
Definition at line 131 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.441 std::\_\_detail::Insert<\_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_↵Traits, \_Constant\_iterators, \_Unique\_keys > Struct Template Reference

Inheritance diagram for std::\_\_detail::Insert<\_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_Rehash↵Policy, \_Traits, \_Constant\_iterators, \_Unique\_keys >:



### 4.441.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits, bool _Constant_iterators = _Traits::__constant_iterators::value, bool
_Unique_keys = _Traits::__unique_keys::value> struct std::__detail::Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2,
_Hash, _RehashPolicy, _Traits, _Constant_iterators, _Unique_keys >
```

Primary class template `_Insert`.

Select insert member functions appropriate to `_Hashtable` policy choices.

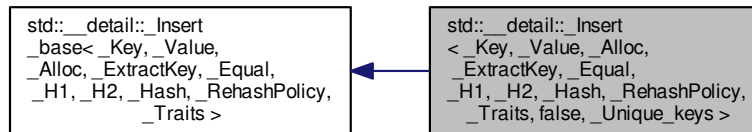
Definition at line 661 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.442 `std::__detail::Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false, _Unique_keys >` Struct Template Reference

Inheritance diagram for `std::__detail::Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false, _Unique_keys >`:



#### Public Types

- using `__base_type` = `_Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`
- using `__hashtable` = `typename __base_type::__hashtable`
- using `__hashtable_base` = `_Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >`
- using `__iconv_type` = `typename __base_type::__iconv_type`
- using `__ireturn_type` = `typename __base_type::__ireturn_type`
- template<typename `_Pair` > using `__is_cons` = `std::is_constructible< value_type, _Pair && >`
- using `__unique_keys` = `typename __base_type::__unique_keys`
- template<typename `_Pair` > using `__IFcons` = `std::enable_if< __is_cons< _Pair >::value >`
- template<typename `_Pair` > using `__IFconsp` = `typename __IFcons< _Pair >::type`
- using `const_iterator` = `typename __base_type::const_iterator`
- using `iterator` = `typename __base_type::iterator`
- using `size_type` = `typename __hashtable_base::size_type`
- using `value_type` = `typename __base_type::value_type`

#### Public Member Functions

- [\\_\\_hashtable](#) & [\\_M\\_conjure\\_hashtable](#) ()
- `__ireturn_type insert` (const value\_type &\_\_v)
- iterator `insert` (const\_iterator, const value\_type &\_\_v)
- void `insert` (initializer\_list< value\_type > \_\_l)
- template<typename \_InputIterator > void `insert` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_Pair, typename = \_IFconsp<\_Pair>> \_\_ireturn\_type `insert` (\_Pair &&\_\_v)
- template<typename \_Pair, typename = \_IFconsp<\_Pair>> iterator `insert` (const\_iterator, \_Pair &&\_\_v)

#### 4.442.1 Detailed Description

template<typename \_Key, typename \_Value, typename \_Alloc, typename \_ExtractKey, typename \_Equal, typename \_H1, typename \_H2, typename \_Hash, typename \_RehashPolicy, typename \_Traits, bool \_Unique\_keys> struct std::\_\_detail::Insert<\_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, false, \_Unique\_keys >

Specialization.

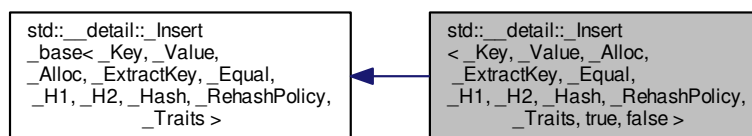
Definition at line 736 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.443 `std::__detail::Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, false >` Struct Template Reference

Inheritance diagram for `std::__detail::Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, false >`:



#### Public Types

- using `__base_type` = [\\_Insert\\_base](#)<\_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits >
- using `__hashtable` = typename `__base_type::__hashtable`
- using `__hashtable_base` = [\\_Hashtable\\_base](#)<\_Key, \_Value, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_Traits >
- using `__iconv_type` = typename `__hashtable_base::__iconv_type`
- using `__ireturn_type` = typename `__hashtable_base::__ireturn_type`
- using `__unique_keys` = typename `__base_type::__unique_keys`
- using `const_iterator` = typename `__base_type::const_iterator`



- using **iterator** = typename \_\_base\_type::iterator
- using **size\_type** = typename \_\_hashtable\_base::size\_type
- using **value\_type** = typename \_\_base\_type::value\_type

#### Public Member Functions

- [\\_\\_hashtable](#) & **\_M\_conjure\_hashtable** ()
- **\_\_ireturn\_type insert** (const value\_type &\_\_v)
- iterator **insert** (const\_iterator, const value\_type &\_\_v)
- void **insert** (initializer\_list< value\_type > \_\_l)
- template<typename \_\_InputIterator > void **insert** (\_\_InputIterator \_\_first, \_\_InputIterator \_\_last)
- iterator **insert** (value\_type &&\_\_v)
- iterator **insert** (const\_iterator, value\_type &&\_\_v)

#### 4.443.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits> struct std::__detail::Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal,
_H1, _H2, _Hash, _RehashPolicy, _Traits, true, false >
```

Specialization.

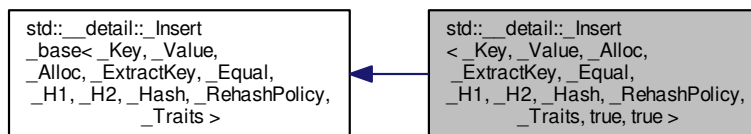
Definition at line 702 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.444 std::\_\_detail::Insert< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, true, true > Struct Template Reference

Inheritance diagram for std::\_\_detail::Insert< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, true, true >:



#### Public Types

- using **\_\_base\_type** = [\\_Insert\\_base](#)< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits >
- using **\_\_hashtable** = typename [\\_\\_base\\_type::\\_\\_hashtable](#)

- using `__hashtable_base` = `_Hashtable_base<_Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >`
- using `__iconv_type` = `typename __hashtable_base::__iconv_type`
- using `__ireturn_type` = `typename __hashtable_base::__ireturn_type`
- using `__unique_keys` = `typename __base_type::__unique_keys`
- using `const_iterator` = `typename __base_type::const_iterator`
- using `iterator` = `typename __base_type::iterator`
- using `size_type` = `typename __hashtable_base::size_type`
- using `value_type` = `typename __base_type::value_type`

#### Public Member Functions

- `__hashtable` & `_M_conjure_hashtable` ()
- `__ireturn_type insert` (const `value_type` &\_\_v)
- `iterator insert` (const `iterator`, const `value_type` &\_\_v)
- void `insert` (initializer\_list< `value_type` > \_\_l)
- template<typename `_InputIterator` > void `insert` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `std::pair`< `iterator`, bool > `insert` (`value_type` &&\_\_v)
- `iterator insert` (const `iterator`, `value_type` &&\_\_v)

#### 4.444.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits> struct std::__detail::Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal,
_H1, _H2, _Hash, _RehashPolicy, _Traits, true, true >
```

Specialization.

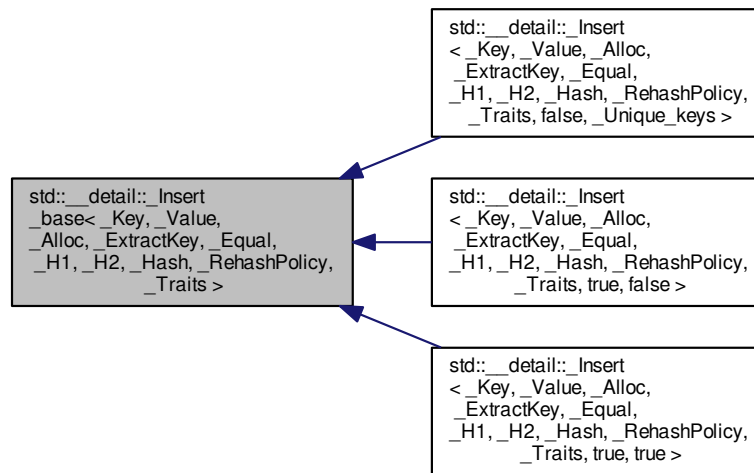
Definition at line 668 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.445 `std::__detail::_Insert_base<_Key,_Value,_Alloc,_ExtractKey,_Equal,_H1,_H2,_Hash,_RehashPolicy,_Traits>` Struct Template Reference

Inheritance diagram for `std::__detail::_Insert_base<_Key,_Value,_Alloc,_ExtractKey,_Equal,_H1,_H2,_Hash,_RehashPolicy,_Traits>`:



#### Public Types

- using `__hashtable` = `_Hashtable<_Key,_Value,_Alloc,_ExtractKey,_Equal,_H1,_H2,_Hash,_RehashPolicy,_Traits>`
- using `__hashtable_base` = `_Hashtable_base<_Key,_Value,_ExtractKey,_Equal,_H1,_H2,_Hash,_Traits>`
- using `__iconv_type` = `typename __hashtable_base::__iconv_type`
- using `__ireturn_type` = `typename __hashtable_base::__ireturn_type`
- using `__unique_keys` = `typename __hashtable_base::__unique_keys`
- using `const_iterator` = `typename __hashtable_base::const_iterator`
- using `iterator` = `typename __hashtable_base::iterator`
- using `size_type` = `typename __hashtable_base::size_type`
- using `value_type` = `typename __hashtable_base::value_type`

#### Public Member Functions

- `__hashtable` & `_M_conjure_hashtable` ()
- `__ireturn_type insert` (const `value_type` &\_\_v)
- `iterator insert` (const `iterator`, const `value_type` &\_\_v)
- void `insert` (initializer\_list< `value_type` > \_\_l)
- template<typename `_InputIterator` > void `insert` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)

## 4.445.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits> struct std::__detail::_Insert_base< _Key, _Value, _Alloc, _ExtractKey,
_Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >
```

Primary class template `_Insert_base`.

insert member functions appropriate to all `_Hashtables`.

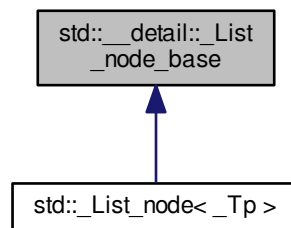
Definition at line 577 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.446 std::\_\_detail::\_List\_node\_base Struct Reference

Inheritance diagram for `std::__detail::_List_node_base`:



## Public Member Functions

- void **\_M\_hook** ([\\_List\\_node\\_base](#) \*const \_\_position) noexcept
- void **\_M\_reverse** () noexcept
- void **\_M\_transfer** ([\\_List\\_node\\_base](#) \*const \_\_first, [\\_List\\_node\\_base](#) \*const \_\_last) noexcept
- void **\_M\_unhook** () noexcept

## Static Public Member Functions

- static void **swap** ([\\_List\\_node\\_base](#) &\_\_x, [\\_List\\_node\\_base](#) &\_\_y) noexcept

## Public Attributes

- [\\_List\\_node\\_base](#) \* **\_M\_next**
- [\\_List\\_node\\_base](#) \* **\_M\_prev**

#### 4.446.1 Detailed Description

Common part of a node in the list.

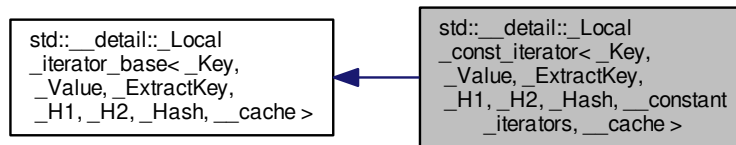
Definition at line 77 of file `stl_list.h`.

The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

#### 4.447 `std::__detail::_Local_const_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >` Struct Template Reference

Inheritance diagram for `std::__detail::_Local_const_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >`:



#### Public Types

- typedef `std::ptrdiff_t` **difference\_type**
- typedef [std::forward\\_iterator\\_tag](#) **iterator\_category**
- typedef `const _Value *` **pointer**
- typedef `const _Value &` **reference**
- typedef `_Value` **value\_type**

#### Public Member Functions

- **\_Local\_const\_iterator** (`const __hash_code_base &__base, \_Hash\_node< _Value, __cache > *__p, std::size_t __t __bkt, std::size_t __bkt_count`)
- **\_Local\_const\_iterator** (`const \_Local\_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache > &__x`)
- reference **operator\*** () const
- [\\_Local\\_const\\_iterator](#) & **operator++** ()
- [\\_Local\\_const\\_iterator](#) **operator++** (int)
- pointer **operator->** () const

#### 4.447.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __constant_iterators, bool __cache>struct std::__detail::__Local_const_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >
```

local const\_iterators

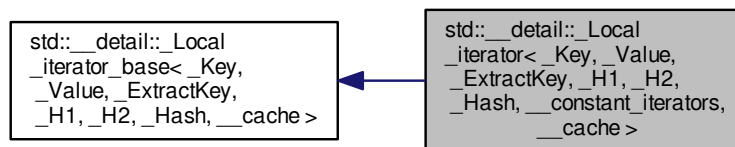
Definition at line 1334 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

4.448 `std::__detail::__Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >` Struct Template Reference

Inheritance diagram for `std::__detail::__Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >`:



#### Public Types

- typedef `std::ptrdiff_t` **difference\_type**
- typedef [std::forward\\_iterator\\_tag](#) **iterator\_category**
- typedef `std::conditional< __constant_iterators, const _Value *, _Value * >::type` **pointer**
- typedef `std::conditional< __constant_iterators, const _Value &, _Value & >::type` **reference**
- typedef `_Value` **value\_type**

#### Public Member Functions

- **\_Local\_iterator** (`const __hash_code_base &__base, \_Hash\_node< _Value, __cache > *__p, std::size_t __bkt, std::size_t __bkt_count`)
- reference **operator\*** () const
- [\\_Local\\_iterator](#) & **operator++** ()
- [\\_Local\\_iterator](#) **operator++** (int)
- pointer **operator->** () const

#### 4.448.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __↵
constant_iterators, bool __cache>struct std::__detail::_Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_↵
iterators, __cache >
```

local iterators

Definition at line 1279 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.449 `std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >` Struct Template Reference

##### 4.449.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache_↵
hash_code>struct std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >
```

Primary class template `_Local_iterator_base`.

Base class for local iterators, used to iterate within a bucket but not between buckets.

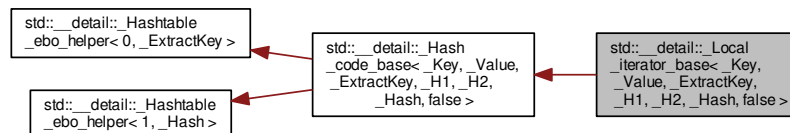
Definition at line 882 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.450 `std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >` Struct Template Reference

Inheritance diagram for `std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >`:



##### Public Member Functions

- `_Local_iterator_base` (const [\\_\\_hash\\_code\\_base](#) &\_\_base, [\\_Hash\\_node](#)< \_Value, false > \*\_\_p, std::size\_t \_\_bkt, std::size\_t \_\_bkt\_count)
- `void _M_incr ()`

## Public Attributes

- `std::size_t _M_bucket`
- `std::size_t _M_bucket_count`
- `_Hash_node<_Value, false > * _M_cur`

## Protected Types

- using `__hash_code_base = _Hash_code_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >`

## Private Types

- typedef void \* `__hash_code`
- typedef `_Hash_node<_Value, false >` `__node_type`

## Private Member Functions

- `std::size_t _M_bucket_index (const _Key &__k, __hash_code, std::size_t __n) const`
- `std::size_t _M_bucket_index (const __node_type * __p, std::size_t __n) const`
- `void _M_copy_code (__node_type *, const __node_type *) const`
- `const _ExtractKey & _M_extract () const`
- `_ExtractKey & _M_extract ()`
- `__hash_code _M_hash_code (const _Key &__key) const`
- `const _Hash & _M_ranged_hash () const`
- `_Hash & _M_ranged_hash ()`
- `void _M_store_code (__node_type *, __hash_code) const`
- `void _M_swap (_Hash_code_base &__x)`

## 4.450.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash> struct std::__detail::__Local_iterator_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >
```

Specialization.

Definition at line 1223 of file `hashtable_policy.h`.

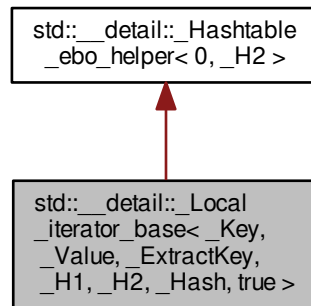
The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)



#### 4.451 `std::__detail::_Local_iterator_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >` Struct Template Reference

Inheritance diagram for `std::__detail::_Local_iterator_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >`:



##### Public Member Functions

- `_Local_iterator_base` (const [\\_\\_hash\\_code\\_base](#) &\_\_base, [\\_Hash\\_node](#)<\_Value, true > \*\_\_p, std::size\_t \_\_bkt, std::size\_t \_\_bkt\_count)
- `void _M_incr ()`

##### Public Attributes

- `std::size_t _M_bucket`
- `std::size_t _M_bucket_count`
- [\\_Hash\\_node](#)<\_Value, true > \* `_M_cur`

##### Protected Types

- using `__base_type` = [\\_Hashtable\\_ebo\\_helper](#)<0, \_H2 >
- using `__hash_code_base` = [\\_Hash\\_code\\_base](#)<\_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, true >

##### 4.451.1 Detailed Description

`template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash>struct std::__detail::_Local_iterator_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >`

Specialization.

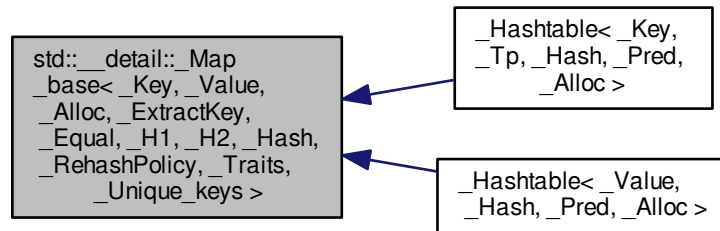
Definition at line 1184 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

4.452 `std::__detail::Map_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >` Struct Template Reference

Inheritance diagram for `std::__detail::Map_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >`:



#### 4.452.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename
_H2, typename _Hash, typename _RehashPolicy, typename _Traits, bool _Unique_keys = _Traits::__unique_keys::value> struct std::__
__detail::Map_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >
```

Primary class template `_Map_base`.

If the hashtable has a value type of the form `pair<T1, T2>` and a key extraction policy (`_ExtractKey`) that returns the first part of the pair, the hashtable gets a `mapped_type` typedef. If it satisfies those criteria and also has unique keys, then it also gets an `operator[]`.

Definition at line 424 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

4.453 `std::__detail::Map_base<_Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >` Struct Template Reference

#### Public Types

- using **`mapped_type`** = `typename std::tuple_element< 1, _Pair >::type`

#### 4.453.1 Detailed Description

```
template<typename _Key, typename _Pair, typename _Alloc, typename _Equal, typename _H1, typename _H2, typename _Hash, type-
name _RehashPolicy, typename _Traits> struct std::__detail::Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash,
_RehashPolicy, _Traits, false >
```

Partial specialization, `__unique_keys` set to false.

Definition at line 430 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.454 `std::__detail::Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >` Struct Template Reference

##### Public Types

- using `iterator` = `typename __hashtable_base::iterator`
- using `key_type` = `typename __hashtable_base::key_type`
- using `mapped_type` = `typename std::tuple_element< 1, _Pair >::type`

##### Public Member Functions

- `mapped_type & at (const key_type &__k)`
- `const mapped_type & at (const key_type &__k) const`
- `mapped_type & operator[] (const key_type &__k)`
- `mapped_type & operator[] (key_type &&__k)`

##### 4.454.1 Detailed Description

```
template<typename _Key, typename _Pair, typename _Alloc, typename _Equal, typename _H1, typename _H2, typename _Hash, type-
name _RehashPolicy, typename _Traits> struct std::__detail::Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash,
_RehashPolicy, _Traits, true >
```

Partial specialization, `__unique_keys` set to true.

Definition at line 440 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.455 `std::__detail::Mod_range_hashing` Struct Reference

##### Public Types

- `typedef std::size_t first_argument_type`
- `typedef std::size_t result_type`
- `typedef std::size_t second_argument_type`

##### Public Member Functions

- `result_type operator() (first_argument_type __num, second_argument_type __den) const`

## 4.455.1 Detailed Description

Default range hashing function: use division to fold a large number into the range [0, N).

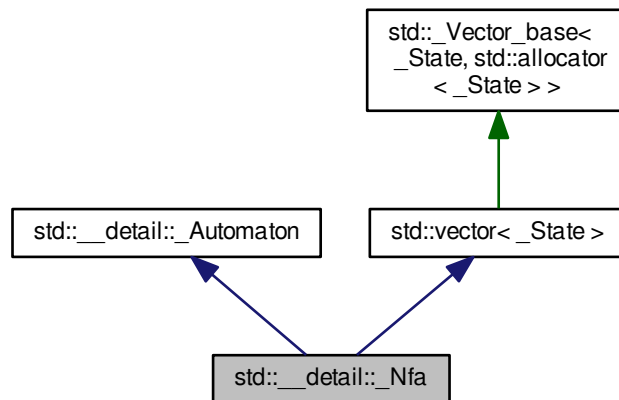
Definition at line 337 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.456 std::\_\_detail::\_Nfa Class Reference

Inheritance diagram for std::\_\_detail::\_Nfa:



## Public Types

- typedef [regex\\_constants::syntax\\_option\\_type](#) **\_FlagT**
- typedef unsigned int **\_SizeT**
- typedef [\\_State](#) **\_StateT**
- typedef [std::allocator<\\_State>](#) **allocator\_type**
- typedef [\\_\\_gnu\\_cxx::\\_\\_normal\\_iterator< const\\_pointer, vector >](#) **const\_iterator**
- typedef [\\_Alloc\\_traits::const\\_pointer](#) **const\_pointer**
- typedef [\\_Alloc\\_traits::const\\_reference](#) **const\_reference**
- typedef [std::reverse\\_iterator< const\\_iterator >](#) **const\_reverse\_iterator**
- typedef ptrdiff\_t **difference\_type**
- typedef [\\_\\_gnu\\_cxx::\\_\\_normal\\_iterator< pointer, vector >](#) **iterator**
- typedef [\\_Base::pointer](#) **pointer**
- typedef [\\_Alloc\\_traits::reference](#) **reference**
- typedef [std::reverse\\_iterator< iterator >](#) **reverse\_iterator**
- typedef size\_t **size\_type**
- typedef [\\_State](#) **value\_type**

## Public Member Functions

- `_Nfa` (`_FlagT __f`)
- `const _StateSet & _M_final_states` () const
- `_StateldT _M_insert_accept` ()
- `_StateldT _M_insert_alt` (`_StateldT __next`, `_StateldT __alt`)
- `_StateldT _M_insert_matcher` (`_Matcher __m`)
- `_StateldT _M_insert_subexpr_begin` (const `_Tagger &__t`)
- `_StateldT _M_insert_subexpr_end` (unsigned int `__i`, const `_Tagger &__t`)
- `_FlagT _M_options` () const
- `_StateldT _M_start` () const
- `_SizeT _M_sub_count` () const
- void `assign` (size\_type `__n`, const value\_type & `__val`)
- void `assign` (`_InputIterator __first`, `_InputIterator __last`)
- void `assign` (initializer\_list< value\_type > `__l`)
- reference `at` (size\_type `__n`)
- const\_reference `at` (size\_type `__n`) const
- reference `back` ()
- const\_reference `back` () const
- iterator `begin` () noexcept
- const\_iterator `begin` () const noexcept
- size\_type `capacity` () const noexcept
- const\_iterator `cbegin` () const noexcept
- const\_iterator `cend` () const noexcept
- void `clear` () noexcept
- `const_reverse_iterator` `crbegin` () const noexcept
- `const_reverse_iterator` `crend` () const noexcept
- `_State * data` () noexcept
- const `_State * data` () const noexcept
- iterator `emplace` (iterator `__position`, `_Args &&... __args`)
- void `emplace_back` (`_Args &&... __args`)
- bool `empty` () const noexcept
- iterator `end` () noexcept
- const\_iterator `end` () const noexcept
- iterator `erase` (iterator `__position`)
- iterator `erase` (iterator `__first`, iterator `__last`)
- reference `front` ()
- const\_reference `front` () const
- iterator `insert` (iterator `__position`, const value\_type & `__x`)
- iterator `insert` (iterator `__position`, value\_type && `__x`)
- void `insert` (iterator `__position`, initializer\_list< value\_type > `__l`)
- void `insert` (iterator `__position`, size\_type `__n`, const value\_type & `__x`)
- void `insert` (iterator `__position`, `_InputIterator __first`, `_InputIterator __last`)
- size\_type `max_size` () const noexcept
- reference `operator[]` (size\_type `__n`)
- const\_reference `operator[]` (size\_type `__n`) const
- void `pop_back` ()
- void `push_back` (const value\_type & `__x`)
- void `push_back` (value\_type && `__x`)
- `reverse_iterator` `rbegin` () noexcept
- `const_reverse_iterator` `rbegin` () const noexcept

- [reverse\\_iterator rend](#) () noexcept
- [const\\_reverse\\_iterator rend](#) () const noexcept
- void [reserve](#) (size\_type \_\_n)
- void [resize](#) (size\_type \_\_new\_size)
- void [resize](#) (size\_type \_\_new\_size, const value\_type &\_\_x)
- void [shrink\\_to\\_fit](#) ()
- size\_type [size](#) () const noexcept
- void [swap](#) ([vector](#) &\_\_x) noexcept([\\_Alloc\\_traits::\\_S\\_nothrow\\_swap](#)())

#### Protected Member Functions

- pointer [\\_M\\_allocate](#) (size\_t \_\_n)
- pointer [\\_M\\_allocate\\_and\\_copy](#) (size\_type \_\_n, [\\_ForwardIterator](#) \_\_first, [\\_ForwardIterator](#) \_\_last)
- void [\\_M\\_assign\\_aux](#) ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, [std::input\\_iterator\\_tag](#))
- void [\\_M\\_assign\\_aux](#) ([\\_ForwardIterator](#) \_\_first, [\\_ForwardIterator](#) \_\_last, [std::forward\\_iterator\\_tag](#))
- void [\\_M\\_assign\\_dispatch](#) ([\\_Integer](#) \_\_n, [\\_Integer](#) \_\_val, \_\_true\_type)
- void [\\_M\\_assign\\_dispatch](#) ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, \_\_false\_type)
- size\_type [\\_M\\_check\\_len](#) (size\_type \_\_n, const char \*\_\_s) const
- void [\\_M\\_deallocate](#) (pointer \_\_p, size\_t \_\_n)
- void [\\_M\\_default\\_append](#) (size\_type \_\_n)
- void [\\_M\\_default\\_initialize](#) (size\_type \_\_n)
- void [\\_M\\_emplace\\_back\\_aux](#) ([\\_Args](#) &&...\_\_args)
- void [\\_M\\_erase\\_at\\_end](#) (pointer \_\_pos)
- void [\\_M\\_fill\\_assign](#) (size\_type \_\_n, const value\_type &\_\_val)
- void [\\_M\\_fill\\_initialize](#) (size\_type \_\_n, const value\_type &\_\_value)
- void [\\_M\\_fill\\_insert](#) (iterator \_\_pos, size\_type \_\_n, const value\_type &\_\_x)
- [\\_Tp\\_alloc\\_type](#) & [\\_M\\_get\\_Tp\\_allocator](#) () noexcept
- const [\\_Tp\\_alloc\\_type](#) & [\\_M\\_get\\_Tp\\_allocator](#) () const noexcept
- void [\\_M\\_initialize\\_dispatch](#) ([\\_Integer](#) \_\_n, [\\_Integer](#) \_\_value, \_\_true\_type)
- void [\\_M\\_initialize\\_dispatch](#) ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, \_\_false\_type)
- void [\\_M\\_insert\\_aux](#) (iterator \_\_position, [\\_Args](#) &&...\_\_args)
- void [\\_M\\_insert\\_dispatch](#) (iterator \_\_pos, [\\_Integer](#) \_\_n, [\\_Integer](#) \_\_val, \_\_true\_type)
- void [\\_M\\_insert\\_dispatch](#) (iterator \_\_pos, [\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, \_\_false\_type)
- void [\\_M\\_range\\_check](#) (size\_type \_\_n) const
- void [\\_M\\_range\\_initialize](#) ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, [std::input\\_iterator\\_tag](#))
- void [\\_M\\_range\\_initialize](#) ([\\_ForwardIterator](#) \_\_first, [\\_ForwardIterator](#) \_\_last, [std::forward\\_iterator\\_tag](#))
- void [\\_M\\_range\\_insert](#) (iterator \_\_pos, [\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, [std::input\\_iterator\\_tag](#))
- void [\\_M\\_range\\_insert](#) (iterator \_\_pos, [\\_ForwardIterator](#) \_\_first, [\\_ForwardIterator](#) \_\_last, [std::forward\\_iterator\\_tag](#))
- bool [\\_M\\_shrink\\_to\\_fit](#) ()
- [allocator\\_type](#) [get\\_allocator](#) () const noexcept

#### Protected Attributes

- [\\_Vector\\_impl](#) [\\_M\\_impl](#)

#### 4.456.1 Detailed Description

struct \_Nfa

A collection of all states making up an NFA.

An NFA is a 4-tuple  $M = (K, S, s, F)$ , where  $K$  is a finite set of states,  $S$  is the alphabet of the NFA,  $s$  is the initial state,  $F$  is a set of final (accepting) states.

This NFA class is templated on  $S$ , a type that will hold values of the underlying alphabet (without regard to semantics of that alphabet). The other elements of the tuple are generated during construction of the NFA and are available through accessor member functions.

Definition at line 269 of file regex\_nfa.h.

#### 4.456.2 Member Function Documentation

**4.456.2.1** pointer `std::vector<_State, std::allocator<_State>>::M_allocate_and_copy ( size_type __n, ForwardIterator __first, ForwardIterator __last )` [inline], [protected], [inherited]

Memory expansion handler. Uses the member allocation function to obtain  $n$  bytes of memory, and then copies [first,last) into it.

Definition at line 1135 of file stl\_vector.h.

**4.456.2.2** void `std::vector<_State, std::allocator<_State>>::M_range_check ( size_type __n ) const` [inline], [protected], [inherited]

Safety check used only from at().

Definition at line 791 of file stl\_vector.h.

References `std::vector<_Tp, _Alloc>::size()`.

**4.456.2.3** void `std::vector<_State, std::allocator<_State>>::assign ( size_type __n, const value_type & __val )` [inline], [inherited]

Assigns a given value to a vector.

Parameters

<code>__n</code>	Number of elements to be assigned.
<code>__val</code>	Value to be assigned.

This function fills a vector with `__n` copies of the given value. Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 479 of file stl\_vector.h.

**4.456.2.4** void `std::vector<_State, std::allocator<_State>>::assign ( InputIterator __first, InputIterator __last )` [inline], [inherited]

Assigns a range to a vector.

Parameters

<code>__first</code>	An input iterator.
----------------------	--------------------

<code>__last</code>	An input iterator.
---------------------	--------------------

This function fills a vector with copies of the elements in the range [`__first`,`__last`).

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 498 of file `stl_vector.h`.

**4.456.2.5** `void std::vector<_State, std::allocator<_State>>::assign ( initializer_list<value_type> & __l )`  
`[inline], [inherited]`

Assigns an initializer list to a vector.

#### Parameters

<code>__l</code>	An initializer_list.
------------------	----------------------

This function fills a vector with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 524 of file `stl_vector.h`.

References `std::vector<_Tp, _Alloc>::assign()`.

**4.456.2.6** `reference std::vector<_State, std::allocator<_State>>::at ( size_type __n )` `[inline]`,  
`[inherited]`

Provides access to the data contained in the vector.

#### Parameters

<code>__n</code>	The index of the element for which data should be accessed.
------------------	---

#### Returns

Read/write reference to data.

#### Exceptions

<code>std::out_of_range</code>	If <code>__n</code> is an invalid index.
--------------------------------	--

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws `out_of_range` if the check fails.

Definition at line 810 of file `stl_vector.h`.

References `std::vector<_Tp, _Alloc>::_M_range_check()`.

**4.456.2.7** `const_reference std::vector<_State, std::allocator<_State>>::at ( size_type __n ) const` `[inline]`,  
`[inherited]`

Provides access to the data contained in the vector.

#### Parameters

<code>__n</code>	The index of the element for which data should be accessed.
------------------	---

#### Returns

Read-only (constant) reference to data.



## Exceptions

<i>std::out_of_range</i>	If <i>__n</i> is an invalid index.
--------------------------	------------------------------------

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws *out\_of\_range* if the check fails.

Definition at line 828 of file *stl\_vector.h*.

References *std::vector<\_Tp, \_Alloc>::M\_range\_check()*.

#### 4.456.2.8 reference *std::vector<\_State, std::allocator<\_State>>::back ( )* [inline], [inherited]

Returns a read/write reference to the data at the last element of the vector.

Definition at line 855 of file *stl\_vector.h*.

References *std::vector<\_Tp, \_Alloc>::end()*.

#### 4.456.2.9 const\_reference *std::vector<\_State, std::allocator<\_State>>::back ( ) const* [inline], [inherited]

Returns a read-only (constant) reference to the data at the last element of the vector.

Definition at line 863 of file *stl\_vector.h*.

References *std::vector<\_Tp, \_Alloc>::end()*.

#### 4.456.2.10 iterator *std::vector<\_State, std::allocator<\_State>>::begin ( )* [inline], [noexcept], [inherited]

Returns a read/write iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 538 of file *stl\_vector.h*.

#### 4.456.2.11 const\_iterator *std::vector<\_State, std::allocator<\_State>>::begin ( ) const* [inline], [noexcept], [inherited]

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 547 of file *stl\_vector.h*.

#### 4.456.2.12 size\_type *std::vector<\_State, std::allocator<\_State>>::capacity ( ) const* [inline], [noexcept], [inherited]

Returns the total number of elements that the vector can hold before needing to allocate more memory.

Definition at line 725 of file *stl\_vector.h*.

#### 4.456.2.13 const\_iterator *std::vector<\_State, std::allocator<\_State>>::cbegin ( ) const* [inline], [noexcept], [inherited]

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 611 of file *stl\_vector.h*.

**4.456.2.14** `const_iterator std::vector<_State, std::allocator<_State>>::cend( ) const` `[inline]`,  
`[noexcept]`, `[inherited]`

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 620 of file `stl_vector.h`.

**4.456.2.15** `void std::vector<_State, std::allocator<_State>>::clear( )` `[inline]`, `[noexcept]`,  
`[inherited]`

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1125 of file `stl_vector.h`.

**4.456.2.16** `const_reverse_iterator std::vector<_State, std::allocator<_State>>::crbegin( ) const` `[inline]`,  
`[noexcept]`, `[inherited]`

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 629 of file `stl_vector.h`.

References `std::vector<_Tp, _Alloc>::end()`.

**4.456.2.17** `const_reverse_iterator std::vector<_State, std::allocator<_State>>::crend( ) const` `[inline]`,  
`[noexcept]`, `[inherited]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 638 of file `stl_vector.h`.

References `std::vector<_Tp, _Alloc>::begin()`.

**4.456.2.18** `_State* std::vector<_State, std::allocator<_State>>::data( )` `[inline]`, `[noexcept]`,  
`[inherited]`

Returns a pointer such that `[data(), data() + size())` is a valid range. For a non-empty vector, `data() == &front()`.

Definition at line 878 of file `stl_vector.h`.

References `std::__addressof()`, and `std::vector<_Tp, _Alloc>::front()`.

**4.456.2.19** `iterator std::vector<_State, std::allocator<_State>>::emplace( iterator __position, _Args&&... __args )`  
`[inherited]`

Inserts an object in vector before specified iterator.

#### Parameters

<code>__position</code>	An iterator into the vector.
<code>__args</code>	Arguments.

#### Returns

An iterator that points to the inserted data.

This function will insert an object of type `T` constructed with `T(std::forward<Args>(args)...) before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using`

std::list.

**4.456.2.20** `bool std::vector<_State, std::allocator<_State>>::empty( ) const` `[inline], [noexcept], [inherited]`

Returns true if the vector is empty. (Thus begin() would equal end().)

Definition at line 734 of file stl\_vector.h.

References std::vector<\_Tp, \_Alloc>::begin(), and std::vector<\_Tp, \_Alloc>::end().

**4.456.2.21** `iterator std::vector<_State, std::allocator<_State>>::end( )` `[inline], [noexcept], [inherited]`

Returns a read/write iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 556 of file stl\_vector.h.

**4.456.2.22** `const_iterator std::vector<_State, std::allocator<_State>>::end( ) const` `[inline], [noexcept], [inherited]`

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 565 of file stl\_vector.h.

**4.456.2.23** `iterator std::vector<_State, std::allocator<_State>>::erase( iterator __position )` `[inherited]`

Remove element at given position.

Parameters

<code>__position</code>	Iterator pointing to element to be erased.
-------------------------	--

Returns

An iterator pointing to the next element (or end()).

This function will erase the element at the given position and thus shorten the vector by one.

Note This operation could be expensive and if it is frequently used the user should consider using std::list. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**4.456.2.24** `iterator std::vector<_State, std::allocator<_State>>::erase( iterator __first, iterator __last )` `[inherited]`

Remove a range of elements.

Parameters

<code>__first</code>	Iterator pointing to the first element to be erased.
<code>__last</code>	Iterator pointing to one past the last element to be erased.

Returns

An iterator pointing to the element pointed to by `__last` prior to erasing (or end()).

This function will erase the elements in the range [`__first`, `__last`) and shorten the vector accordingly.

Note This operation could be expensive and if it is frequently used the user should consider using `std::list`. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**4.456.2.25** reference `std::vector<_State, std::allocator<_State>>::front ( )` `[inline]`, `[inherited]`

Returns a read/write reference to the data at the first element of the vector.

Definition at line 839 of file `stl_vector.h`.

References `std::vector<_Tp, _Alloc>::begin()`.

**4.456.2.26** const\_reference `std::vector<_State, std::allocator<_State>>::front ( ) const` `[inline]`, `[inherited]`

Returns a read-only (constant) reference to the data at the first element of the vector.

Definition at line 847 of file `stl_vector.h`.

References `std::vector<_Tp, _Alloc>::begin()`.

**4.456.2.27** iterator `std::vector<_State, std::allocator<_State>>::insert ( iterator __position, const value_type & __x )` `[inherited]`

Inserts given value into vector before specified iterator.

Parameters

<code>__position</code>	An iterator into the vector.
<code>__x</code>	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

**4.456.2.28** iterator `std::vector<_State, std::allocator<_State>>::insert ( iterator __position, value_type && __x )` `[inline]`, `[inherited]`

Inserts given rvalue into vector before specified iterator.

Parameters

<code>__position</code>	An iterator into the vector.
<code>__x</code>	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 988 of file `stl_vector.h`.

References `std::vector<_Tp, _Alloc>::emplace()`, and `std::move()`.

4.456.2.29 `void std::vector<_State, std::allocator<_State>>::insert ( iterator __position, initializer_list< value_type > __l ) [inline],[inherited]`

Inserts an initializer\_list into the vector.

## Parameters

<code>__position</code>	An iterator into the vector.
<code>__l</code>	An initializer_list.

This function will insert copies of the data in the initializer\_list *l* into the vector before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 1005 of file `stl_vector.h`.

References `std::vector<_Tp, _Alloc>::insert()`.

**4.456.2.30** `void std::vector<_State, std::allocator<_State>>::insert ( iterator __position, size_type __n, const value_type & __x ) [inline], [inherited]`

Inserts a number of copies of given data into the vector.

## Parameters

<code>__position</code>	An iterator into the vector.
<code>__n</code>	Number of elements to be inserted.
<code>__x</code>	Data to be inserted.

This function will insert a specified number of copies of the given data before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 1023 of file `stl_vector.h`.

**4.456.2.31** `void std::vector<_State, std::allocator<_State>>::insert ( iterator __position, _InputIterator __first, _InputIterator __last ) [inline], [inherited]`

Inserts a range into the vector.

## Parameters

<code>__position</code>	An iterator into the vector.
<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

This function will insert copies of the data in the range [`__first`,`__last`) into the vector before the location specified by *pos*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 1044 of file `stl_vector.h`.

**4.456.2.32** `size_type std::vector<_State, std::allocator<_State>>::max_size ( ) const [inline], [noexcept], [inherited]`

Returns the `size()` of the largest possible vector.

Definition at line 650 of file `stl_vector.h`.

References `std::allocator_traits<_Tp_alloc_type>::max_size()`.

**4.456.2.33** `reference std::vector<_State, std::allocator<_State>>::operator[] ( size_type __n ) [inline], [inherited]`

Subscript access to the data contained in the vector.

**Parameters**

<code>__n</code>	The index of the element for which data should be accessed.
------------------	---

**Returns**

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 770 of file `stl_vector.h`.

**4.456.2.34** `const_reference std::vector<_State, std::allocator<_State>>::operator[] ( size_type __n ) const`  
`[inline], [inherited]`

Subscript access to the data contained in the vector.

**Parameters**

<code>__n</code>	The index of the element for which data should be accessed.
------------------	---

**Returns**

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 785 of file `stl_vector.h`.

**4.456.2.35** `void std::vector<_State, std::allocator<_State>>::pop_back ( ) [inline], [inherited]`

Removes last element.

This is a typical stack operation. It shrinks the vector by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before `pop_back()` is called.

Definition at line 937 of file `stl_vector.h`.

**4.456.2.36** `void std::vector<_State, std::allocator<_State>>::push_back ( const value_type & __x ) [inline], [inherited]`

Add data to the end of the vector.

**Parameters**

<code>__x</code>	Data to be added.
------------------	-------------------

This is a typical stack operation. The function creates an element at the end of the vector and assigns the given data to it. Due to the nature of a vector this operation can be done in constant time if the vector has preallocated space available.

Definition at line 901 of file `stl_vector.h`.

References `std::vector<_Tp, _Alloc>::end()`.

**4.456.2.37** `reverse_iterator std::vector<_State, std::allocator<_State>>::rbegin ( ) [inline], [noexcept], [inherited]`

Returns a read/write reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 574 of file stl\_vector.h.

References `std::vector<_Tp, _Alloc>::end()`.

**4.456.2.38** `const_reverse_iterator std::vector<_State, std::allocator<_State>>::rbegin( ) const` `[inline]`, `[noexcept]`, `[inherited]`

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 583 of file stl\_vector.h.

References `std::vector<_Tp, _Alloc>::end()`.

**4.456.2.39** `reverse_iterator std::vector<_State, std::allocator<_State>>::rend( )` `[inline]`, `[noexcept]`, `[inherited]`

Returns a read/write reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 592 of file stl\_vector.h.

References `std::vector<_Tp, _Alloc>::begin()`.

**4.456.2.40** `const_reverse_iterator std::vector<_State, std::allocator<_State>>::rend( ) const` `[inline]`, `[noexcept]`, `[inherited]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 601 of file stl\_vector.h.

References `std::vector<_Tp, _Alloc>::begin()`.

**4.456.2.41** `void std::vector<_State, std::allocator<_State>>::reserve( size_type __n )` `[inherited]`

Attempt to preallocate enough memory for specified number of elements.

#### Parameters

<code>__n</code>	Number of elements required.
------------------	------------------------------

#### Exceptions

<code>std::length_error</code>	If <code>n</code> exceeds <code>max_size()</code> .
--------------------------------	---

This function attempts to reserve enough memory for the vector to hold the specified number of elements. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the number of elements that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of vector data.

**4.456.2.42** `void std::vector<_State, std::allocator<_State>>::resize( size_type __new_size )` `[inline]`, `[inherited]`

Resizes the vector to the specified number of elements.



## Parameters

<code>__new_size</code>	Number of elements the vector should contain.
-------------------------	---

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise default constructed elements are appended.

Definition at line 664 of file `stl_vector.h`.

References `std::vector<_Tp, _Alloc>::size()`.

**4.456.2.43** `void std::vector<_State, std::allocator<_State>>::resize ( size_type __new_size, const value_type & __x )`  
`[inline], [inherited]`

Resizes the vector to the specified number of elements.

## Parameters

<code>__new_size</code>	Number of elements the vector should contain.
<code>__x</code>	Data with which new elements should be populated.

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise the vector is extended and new elements are populated with given data.

Definition at line 684 of file `stl_vector.h`.

References `std::vector<_Tp, _Alloc>::end()`, `std::vector<_Tp, _Alloc>::insert()`, and `std::vector<_Tp, _Alloc>::size()`.

**4.456.2.44** `void std::vector<_State, std::allocator<_State>>::shrink_to_fit ( )` `[inline], [inherited]`

A non-binding request to reduce `capacity()` to `size()`.

Definition at line 716 of file `stl_vector.h`.

**4.456.2.45** `size_type std::vector<_State, std::allocator<_State>>::size ( ) const` `[inline], [noexcept], [inherited]`

Returns the number of elements in the vector.

Definition at line 645 of file `stl_vector.h`.

**4.456.2.46** `void std::vector<_State, std::allocator<_State>>::swap ( vector<_State> & __x )` `[inline], [noexcept], [inherited]`

Swaps data with another vector.

## Parameters

<code>__x</code>	A vector of the same element and allocator types.
------------------	---

This exchanges the elements between two vectors in constant time. (Three pointers, so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(v1,v2)` will feed to this function.

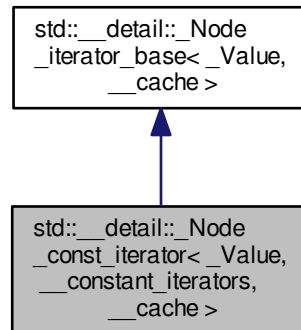
Definition at line 1108 of file `stl_vector.h`.

The documentation for this class was generated from the following file:

- [regex\\_nfa.h](#)

4.457 `std::__detail::_Node_const_iterator< _Value, __constant_iterators, __cache >` Struct Template Reference

Inheritance diagram for `std::__detail::_Node_const_iterator< _Value, __constant_iterators, __cache >`:



## Public Types

- typedef `std::ptrdiff_t` **difference\_type**
- typedef `std::forward_iterator_tag` **iterator\_category**
- typedef `const _Value *` **pointer**
- typedef `const _Value &` **reference**
- typedef `_Value` **value\_type**

## Public Member Functions

- **\_Node\_const\_iterator** (`__node_type *` \_\_p)
- **\_Node\_const\_iterator** (`const _Node_iterator< _Value, __constant_iterators, __cache > &` \_\_x)
- `void` **\_M\_incr** ()
- `reference` **operator\*** () `const`
- `_Node_const_iterator &` **operator++** ()
- `_Node_const_iterator` **operator++** (`int`)
- `pointer` **operator->** () `const`

## Public Attributes

- `__node_type *` **\_M\_cur**

## 4.457.1 Detailed Description

```
template<typename _Value, bool __constant_iterators, bool __cache>struct std::__detail::_Node_const_iterator< _Value, __constant_iterators, __cache >
```

Node const\_iterators, used to iterate through all the hashtable.

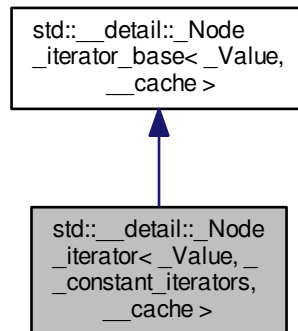
Definition at line 282 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.458 std::\_\_detail::\_Node\_iterator< \_Value, \_\_constant\_iterators, \_\_cache > Struct Template Reference

Inheritance diagram for std::\_\_detail::\_Node\_iterator< \_Value, \_\_constant\_iterators, \_\_cache >:



#### Public Types

- typedef std::ptrdiff\_t **difference\_type**
- typedef [std::forward\\_iterator\\_tag](#) **iterator\_category**
- using **pointer** = typename std::conditional< \_\_constant\_iterators, const \_Value \*, \_Value \* >::type
- using **reference** = typename std::conditional< \_\_constant\_iterators, const \_Value &, \_Value & >::type
- typedef \_Value **value\_type**

#### Public Member Functions

- **\_Node\_iterator** (\_\_node\_type \*\_\_p)
- void **\_M\_incr** ()
- reference **operator\*** () const
- [\\_Node\\_iterator](#) & **operator++** ()
- [\\_Node\\_iterator](#) **operator++** (int)
- pointer **operator->** () const

## Public Attributes

- `__node_type * _M_cur`

## 4.458.1 Detailed Description

```
template<typename _Value, bool __constant_iterators, bool __cache>struct std::__detail::__Node_iterator< _Value, __constant_↵
iterators, __cache >
```

Node iterators, used to iterate through all the hashtable.

Definition at line 231 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.459 std::\_\_detail::\_\_Node\_iterator\_base&lt; \_Value, \_Cache\_hash\_code &gt; Struct Template Reference

## Public Types

- using `__node_type = _Hash_node< _Value, _Cache_hash_code >`

## Public Member Functions

- `_Node_iterator_base ( __node_type * __p)`
- `void _M_incr ()`

## Public Attributes

- `__node_type * _M_cur`

## 4.459.1 Detailed Description

```
template<typename _Value, bool _Cache_hash_code>struct std::__detail::__Node_iterator_base< _Value, _Cache_hash_code >
```

Base class for node iterators.

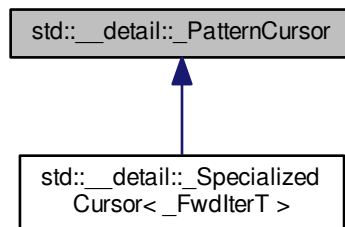
Definition at line 203 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.460 `std::__detail::_PatternCursor` Struct Reference

Inheritance diagram for `std::__detail::_PatternCursor`:



##### Public Member Functions

- virtual bool **\_M\_at\_end** () const =0
- virtual void **\_M\_next** ()=0

##### 4.460.1 Detailed Description

ABC for pattern matching.

Definition at line 44 of file `regex_cursor.h`.

The documentation for this struct was generated from the following file:

- [regex\\_cursor.h](#)

#### 4.461 `std::__detail::_Prime_rehash_policy` Struct Reference

##### Public Types

- enum { **\_S\_n\_primes** }
- typedef `std::size_t` **\_State**

##### Public Member Functions

- **\_Prime\_rehash\_policy** (float \_\_z=1.0)
- `std::size_t` **\_M\_bkt\_for\_elements** (`std::size_t` \_\_n) const
- [std::pair](#)< bool, `std::size_t` > **\_M\_need\_rehash** (`std::size_t` \_\_n\_bkt, `std::size_t` \_\_n\_elt, `std::size_t` \_\_n\_ins) const
- `std::size_t` **\_M\_next\_bkt** (`std::size_t` \_\_n) const
- void **\_M\_reset** (\_State \_\_state)
- \_State **\_M\_state** () const
- float **max\_load\_factor** () const noexcept

## Public Attributes

- float **\_M\_max\_load\_factor**
- std::size\_t **\_M\_next\_resize**

## Static Public Attributes

- static const std::size\_t **\_S\_growth\_factor**

## 4.461.1 Detailed Description

Default value for rehash policy. Bucket size is (usually) the smallest prime that keeps the load factor small enough.

Definition at line 357 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.462 std::\_\_detail::\_RangeMatcher&lt; \_InIterT, \_TraitsT &gt; Struct Template Reference

## Public Types

- typedef \_TraitsT::char\_type **\_CharT**
- typedef [std::basic\\_string](#)< \_CharT > **\_StringT**

## Public Member Functions

- **\_RangeMatcher** (bool \_\_is\_non\_matching, const \_TraitsT & \_\_t=\_TraitsT())
- void **\_M\_add\_char** ( \_CharT \_\_c)
- void **\_M\_add\_character\_class** (const [\\_StringT](#) & \_\_s)
- void **\_M\_add\_collating\_element** (const [\\_StringT](#) & \_\_s)
- void **\_M\_add\_equivalence\_class** (const [\\_StringT](#) & \_\_s)
- void **\_M\_make\_range** ()
- bool **operator()** (const [\\_PatternCursor](#) & \_\_pc) const

## Public Attributes

- bool **\_M\_is\_non\_matching**
- const \_TraitsT & **\_M\_traits**

## 4.462.1 Detailed Description

```
template<typename _InIterT, typename _TraitsT> struct std::__detail::_RangeMatcher< _InIterT, _TraitsT >
```

Matches a character range (bracket expression)

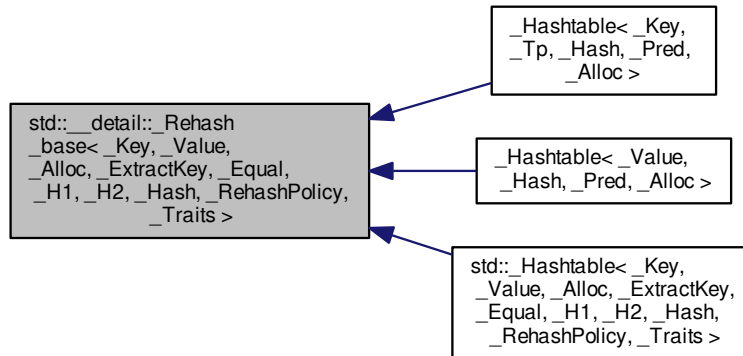
Definition at line 152 of file regex\_nfa.h.

The documentation for this struct was generated from the following file:

- [regex\\_nfa.h](#)

#### 4.463 `std::__detail::_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >` Struct Template Reference

Inheritance diagram for `std::__detail::_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`:



##### 4.463.1 Detailed Description

`template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2, typename _Hash, typename _RehashPolicy, typename _Traits> struct std::__detail::_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`

Primary class template `_Rehash_base`.

Give hashtable the `max_load_factor` functions and reserve iff the rehash policy is `_Prime_rehash_policy`.

Definition at line 788 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.464 `std::__detail::_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _Prime_rehash_policy, _Traits >` Struct Template Reference

##### Public Types

- using `__hashtable` = `_Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _Prime_rehash_policy, _Traits >`

##### Public Member Functions

- float `max_load_factor` () const noexcept
- void `max_load_factor` (float \_\_z)
- void `reserve` (std::size\_t \_\_n)

## 4.464.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _Traits> struct std::__detail::_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash,
_Prime_rehash_policy, _Traits >
```

Specialization.

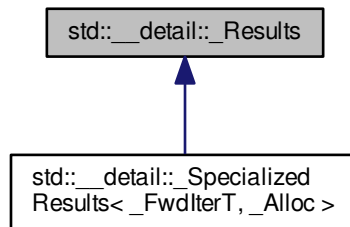
Definition at line 794 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.465 std::\_\_detail::\_Results Struct Reference

Inheritance diagram for std::\_\_detail::\_Results:



## Public Member Functions

- virtual void **\_M\_set\_matched** (int \_\_i, bool \_\_is\_matched)=0
- virtual void **\_M\_set\_pos** (int \_\_i, int \_\_j, const [\\_PatternCursor](#) &\_\_p)=0

## 4.465.1 Detailed Description

Provides a generic facade for a templated match\_results.

Definition at line 77 of file regex\_nfa.h.

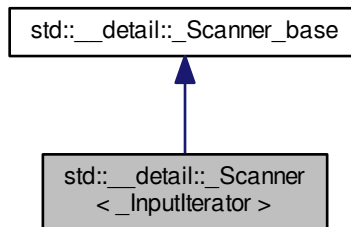
The documentation for this struct was generated from the following file:

- [regex\\_nfa.h](#)



#### 4.466 `std::__detail::_Scanner<_InputIterator>` Class Template Reference

Inheritance diagram for `std::__detail::_Scanner<_InputIterator>`:



##### Public Types

- typedef `std::iterator_traits<_IteratorT>::value_type` **\_CharT**
- typedef const `std::ctype<_CharT>` **\_CtypeT**
- typedef `regex_constants::syntax_option_type` **\_FlagT**
- typedef `_InputIterator` **\_IteratorT**
- typedef unsigned int **\_StateT**
- typedef `std::basic_string<_CharT>` **\_StringT**
- enum **\_TokenT** { `_S_token_anychar`, `_S_token_backref`, `_S_token_bracket_begin`, `_S_token_bracket_end`, `_S_token_inverse_class`, `_S_token_char_class_name`, `_S_token_closure0`, `_S_token_closure1`, `_S_token_collelem_multi`, `_S_token_collelem_single`, `_S_token_collsymbol`, `_S_token_comma`, `_S_token_dash`, `_S_token_dup_count`, `_S_token_eof`, `_S_token_equiv_class_name`, `_S_token_interval_begin`, `_S_token_interval_end`, `_S_token_line_begin`, `_S_token_line_end`, `_S_token_opt`, `_S_token_or`, `_S_token_ord_char`, `_S_token_quoted_char`, `_S_token_subexpr_begin`, `_S_token_subexpr_end`, `_S_token_word_begin`, `_S_token_word_end`, `_S_token_unknown` }

##### Public Member Functions

- **\_Scanner** (`_IteratorT __begin`, `_IteratorT __end`, `_FlagT __flags`, `std::locale __loc`)
- void **\_M\_advance** ()
- **\_TokenT \_M\_token** () const
- const **\_StringT & \_M\_value** () const

##### Static Public Attributes

- static constexpr `_StateT` **\_S\_state\_at\_start**
- static constexpr `_StateT` **\_S\_state\_in\_brace**
- static constexpr `_StateT` **\_S\_state\_in\_bracket**

## 4.466.1 Detailed Description

```
template<typename _InputIterator> class std::__detail::_Scanner< _InputIterator >
```

struct \_Scanner. Scans an input range for regex tokens.

The \_Scanner class interprets the regular expression pattern in the input range passed to its constructor as a sequence of parse tokens passed to the regular expression compiler. The sequence of tokens provided depends on the flag settings passed to the constructor: different regular expression grammars will interpret the same input pattern in syntactically different ways.

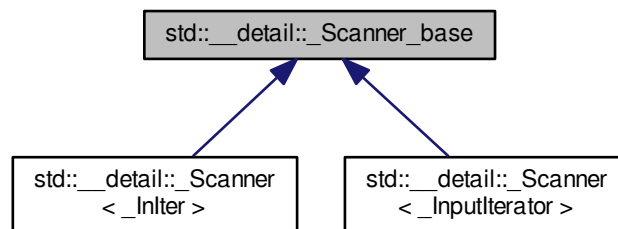
Definition at line 65 of file regex\_compiler.h.

The documentation for this class was generated from the following file:

- [regex\\_compiler.h](#)

## 4.467 std::\_\_detail::\_Scanner\_base Struct Reference

Inheritance diagram for std::\_\_detail::\_Scanner\_base:



## Public Types

- typedef unsigned int **\_StateT**

## Static Public Attributes

- static constexpr \_StateT **\_S\_state\_at\_start**
- static constexpr \_StateT **\_S\_state\_in\_brace**
- static constexpr \_StateT **\_S\_state\_in\_bracket**

## 4.467.1 Detailed Description

Base class for scanner.

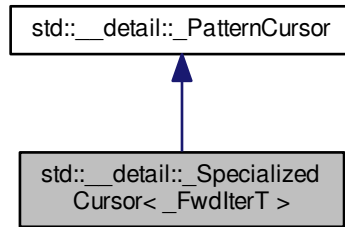
Definition at line 43 of file regex\_compiler.h.

The documentation for this struct was generated from the following file:

- [regex\\_compiler.h](#)

#### 4.468 std::\_\_detail::\_\_SpecializedCursor< \_FwdIterT > Class Template Reference

Inheritance diagram for std::\_\_detail::\_\_SpecializedCursor< \_FwdIterT >:



#### Public Member Functions

- **\_SpecializedCursor** (const \_FwdIterT &\_\_b, const \_FwdIterT \_\_e)
- bool **\_M\_at\_end** () const
- const \_FwdIterT & **\_M\_begin** () const
- std::iterator\_traits< \_FwdIterT >::value\_type **\_M\_current** () const
- const \_FwdIterT & **\_M\_end** () const
- void **\_M\_next** ()
- \_FwdIterT **\_M\_pos** () const

##### 4.468.1 Detailed Description

```
template<typename _FwdIterT> class std::__detail::__SpecializedCursor< _FwdIterT >
```

Provides a cursor into the specific target string.

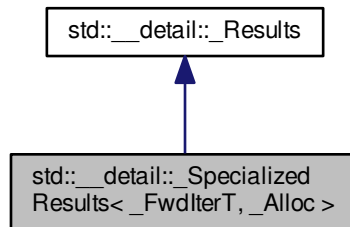
Definition at line 53 of file regex\_cursor.h.

The documentation for this class was generated from the following file:

- [regex\\_cursor.h](#)

## 4.469 std::\_\_detail::\_SpecializedResults&lt; \_FwdIterT, \_Alloc &gt; Class Template Reference

Inheritance diagram for std::\_\_detail::\_SpecializedResults< \_FwdIterT, \_Alloc >:



## Public Member Functions

- **\_SpecializedResults** (const \_Automaton::SizeT \_\_size, const [\\_SpecializedCursor](#)< \_FwdIterT > &\_\_cursor, [match\\_results](#)< \_FwdIterT, \_Alloc > &\_\_m)
- void **\_M\_set\_matched** (int \_\_i, bool \_\_is\_matched)
- void **\_M\_set\_pos** (int \_\_i, int \_\_j, const [\\_PatternCursor](#) &\_\_pc)

## 4.469.1 Detailed Description

```
template<typename _FwdIterT, typename _Alloc>class std::__detail::_SpecializedResults< _FwdIterT, _Alloc >
```

A \_Results facade specialized for wrapping a templated match\_results.

Definition at line 55 of file regex\_grep\_matcher.h.

The documentation for this class was generated from the following file:

- [regex\\_grep\\_matcher.h](#)

## 4.470 std::\_\_detail::\_StartTagger&lt; \_FwdIterT, \_TraitsT &gt; Struct Template Reference

## Public Member Functions

- **\_StartTagger** (int \_\_i)
- void **operator()** (const [\\_PatternCursor](#) &\_\_pc, [\\_Results](#) &\_\_r)

## Public Attributes

- int **\_M\_index**

#### 4.470.1 Detailed Description

```
template<typename _FwdIterT, typename _TraitsT> struct std::__detail::_StartTagger< _FwdIterT, _TraitsT >
```

Start state tag.

Definition at line 88 of file `regex_nfa.h`.

The documentation for this struct was generated from the following file:

- [regex\\_nfa.h](#)

#### 4.471 std::\_\_detail::\_State Struct Reference

##### Public Types

- typedef int **\_OpcodeT**

##### Public Member Functions

- **\_State** (`_OpcodeT __opcode`)
- **\_State** (`const \_Matcher &__m`)
- **\_State** (`_OpcodeT __opcode, unsigned int __s, const \_Tagger &__t`)
- **\_State** (`\_StateIdT __next, \_StateIdT __alt`)

##### Public Attributes

- `\_StateIdT _M_alt`
- `\_Matcher _M_matches`
- `\_StateIdT _M_next`
- `_OpcodeT _M_opcode`
- `unsigned int _M_subexpr`
- `\_Tagger _M_tagger`

#### 4.471.1 Detailed Description

struct `_State`

An individual state in an NFA

In this case a "state" is an entry in the NFA definition coupled with its outgoing transition(s). All states have a single outgoing transition, except for accepting states (which have no outgoing transitions) and alt states, which have two outgoing transitions.

Definition at line 211 of file `regex_nfa.h`.

The documentation for this struct was generated from the following file:

- [regex\\_nfa.h](#)

## 4.472 std::\_\_detail::\_StateSeq Class Reference

## Public Member Functions

- [\\_StateSeq](#) ([\\_Nfa](#) &\_\_ss, [\\_StateIdT](#) \_\_s, [\\_StateIdT](#) \_\_e=[\\_S\\_invalid\\_state\\_id](#))
- [\\_StateSeq](#) (const [\\_StateSeq](#) &\_\_e1, const [\\_StateSeq](#) &\_\_e2)
- [\\_StateSeq](#) (const [\\_StateSeq](#) &\_\_e, [\\_StateIdT](#) \_\_id)
- [\\_StateSeq](#) (const [\\_StateSeq](#) &\_\_rhs)
- void [\\_M\\_append](#) ([\\_StateIdT](#) \_\_id)
- void [\\_M\\_append](#) ([\\_StateSeq](#) &\_\_rhs)
- [\\_StateIdT](#) [\\_M\\_clone](#) ()
- [\\_StateIdT](#) [\\_M\\_front](#) () const
- void [\\_M\\_push\\_back](#) ([\\_StateIdT](#) \_\_id)
- [\\_StateSeq](#) & [operator=](#) (const [\\_StateSeq](#) &\_\_rhs)

## 4.472.1 Detailed Description

Describes a sequence of one or more [\\_State](#), its current start and end(s). This structure contains fragments of an NFA during construction.

Definition at line 352 of file [regex\\_nfa.h](#).

The documentation for this class was generated from the following file:

- [regex\\_nfa.h](#)

## 4.473 std::\_\_exception\_ptr::exception\_ptr Class Reference

## Public Member Functions

- [exception\\_ptr](#) (const [exception\\_ptr](#) &) noexcept
- [exception\\_ptr](#) (nullptr\_t) noexcept
- [exception\\_ptr](#) ([exception\\_ptr](#) &&\_\_o) noexcept
- const class [std::type\\_info](#) \* [\\_\\_cxa\\_exception\\_type](#) () const noexcept [\\_\\_attribute\\_\\_\(\(\\_\\_pure\\_\\_\)\)](#)
- [operator bool](#) () const
- [exception\\_ptr](#) & [operator=](#) (const [exception\\_ptr](#) &) noexcept
- [exception\\_ptr](#) & [operator=](#) ([exception\\_ptr](#) &&\_\_o) noexcept
- void [swap](#) ([exception\\_ptr](#) &) noexcept

## Friends

- bool [operator==](#) (const [exception\\_ptr](#) &, const [exception\\_ptr](#) &) noexcept [\\_\\_attribute\\_\\_\(\(\\_\\_pure\\_\\_\)\)](#)
- [exception\\_ptr](#) [std::current\\_exception](#) () noexcept
- void [std::rethrow\\_exception](#) ([exception\\_ptr](#))

## 4.473.1 Detailed Description

An opaque pointer to an arbitrary exception.

Definition at line 75 of file [exception\\_ptr.h](#).

The documentation for this class was generated from the following file:

- [exception\\_ptr.h](#)

#### 4.474 `std::__has_iterator_category_helper<_Tp>` Class Template Reference

Inherits `__sfinae_types`.

##### Static Public Attributes

- static constexpr bool **value**

##### 4.474.1 Detailed Description

```
template<typename _Tp>class std::__has_iterator_category_helper<_Tp>
```

Traits class for iterators.

This class does nothing but define nested typedefs. The general version simply *forwards* the nested typedefs from the Iterator argument. Specialized versions for pointers and pointers-to-const provide tighter, more correct semantics.

Definition at line 142 of file `stl_iterator_base_types.h`.

The documentation for this class was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

#### 4.475 `std::__parallel::CRandNumber<_MustBeInt>` Struct Template Reference

##### Public Member Functions

- int **operator()** (int \_\_limit)

##### 4.475.1 Detailed Description

```
template<typename _MustBeInt = int>struct std::__parallel::CRandNumber<_MustBeInt>
```

Functor wrapper for `std::rand()`.

Definition at line 1656 of file `algo.h`.

The documentation for this struct was generated from the following file:

- [algo.h](#)

#### 4.476 `std::__profile::map<_Key, _Tp, _Compare, _Allocator>` Class Template Reference

Inherits `map<_Key, _Tp, _Compare, _Allocator>`.

##### Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `_Base::const_iterator` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**

- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `std::pair< const_Key, _Tp >` **value\_type**

## Public Member Functions

- **map** (const `_Compare` &\_\_comp=`_Compare()`, const `_Allocator` &\_\_a=`_Allocator()`)
- template<typename `_InputIterator` , typename = `std::RequireInputIter<_InputIterator>>` **map** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp=`_Compare()`, const `_Allocator` &\_\_a=`_Allocator()`)
- **map** (const `map` &\_\_x)
- **map** (const `_Base` &\_\_x)
- **map** (`map` &&\_\_x) noexcept(is\_nothrow\_copy\_constructible< `_Compare` >::value)
- **map** (initializer\_list< `value_type` > \_\_l, const `_Compare` &\_\_c=`_Compare()`, const `allocator_type` &\_\_a=`allocator_type()`)
- `_Base` & **\_M\_base** () noexcept
- const `_Base` & **\_M\_base** () const noexcept
- `mapped_type` & **at** (const `key_type` &\_\_k)
- const `mapped_type` & **at** (const `key_type` &\_\_k) const
- `iterator` **begin** () noexcept
- const `iterator` **begin** () const noexcept
- const `iterator` **cbegin** () const noexcept
- const `iterator` **end** () const noexcept
- void **clear** () noexcept
- `size_type` **count** (const `key_type` &\_\_x) const
- `const_reverse_iterator` **crbegin** () const noexcept
- `const_reverse_iterator` **crend** () const noexcept
- template<typename... `_Args`> `std::pair< iterator, bool >` **emplace** (`_Args` &&...\_\_args)
- template<typename... `_Args`> `iterator` **emplace\_hint** (const `iterator` \_\_pos, `_Args` &&...\_\_args)
- `iterator` **end** () noexcept
- const `iterator` **end** () const noexcept
- `std::pair< iterator, iterator >` **equal\_range** (const `key_type` &\_\_x)
- `std::pair< const_iterator, const_iterator >` **equal\_range** (const `key_type` &\_\_x) const
- `iterator` **erase** (const `iterator` \_\_position)
- `iterator` **erase** (`iterator` \_\_position)
- `size_type` **erase** (const `key_type` &\_\_x)
- `iterator` **erase** (const `iterator` \_\_first, const `iterator` \_\_last)
- `iterator` **find** (const `key_type` &\_\_x)
- const `iterator` **find** (const `key_type` &\_\_x) const
- `std::pair< iterator, bool >` **insert** (const `value_type` &\_\_x)
- template<typename `_Pair` , typename = `typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>` `std::pair< iterator, bool >` **insert** (`_Pair` &&\_\_x)
- void **insert** (std::initializer\_list< `value_type` > \_\_list)



- iterator **insert** (const\_iterator \_\_position, const [value\\_type](#) &\_\_x)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type> iterator **insert** (const\_iterator \_\_position, \_Pair &&\_\_x)
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>> void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- iterator **lower\_bound** (const key\_type &\_\_x)
- const\_iterator **lower\_bound** (const key\_type &\_\_x) const
- [map](#) & **operator=** (const [map](#) &\_\_x)
- [map](#) & **operator=** ([map](#) &&\_\_x)
- [map](#) & **operator=** (initializer\_list< [value\\_type](#) > \_\_l)
- mapped\_type & **operator[]** (const key\_type &\_\_k)
- mapped\_type & **operator[]** (key\_type &&\_\_k)
- [reverse\\_iterator](#) **rbegin** () noexcept
- [const\\_reverse\\_iterator](#) **rbegin** () const noexcept
- [reverse\\_iterator](#) **rend** () noexcept
- [const\\_reverse\\_iterator](#) **rend** () const noexcept
- void **swap** ([map](#) &\_\_x)
- iterator **upper\_bound** (const key\_type &\_\_x)
- const\_iterator **upper\_bound** (const key\_type &\_\_x) const

#### 4.476.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std::pair<const _Key, _Tp>>>>class std::__profile::map< _Key, _Tp, _Compare, _Allocator >
```

Class std::map wrapper with performance instrumentation.

Definition at line 41 of file profile/map.h.

The documentation for this class was generated from the following file:

- [profile/map.h](#)

#### 4.477 std::\_\_profile::multimap< \_Key, \_Tp, \_Compare, \_Allocator > Class Template Reference

Inherits multimap< \_Key, \_Tp, \_Compare, \_Allocator >.

##### Public Types

- typedef \_Allocator **allocator\_type**
- typedef \_Base::const\_iterator **const\_iterator**
- typedef \_Base::const\_pointer **const\_pointer**
- typedef \_Base::const\_reference **const\_reference**
- typedef \_Base::const\_reverse\_iterator **const\_reverse\_iterator**
- typedef \_Base::difference\_type **difference\_type**
- typedef \_Base::iterator **iterator**
- typedef \_Compare **key\_compare**
- typedef \_Key **key\_type**
- typedef \_Tp **mapped\_type**
- typedef \_Base::pointer **pointer**
- typedef \_Base::reference **reference**

- typedef \_Base::reverse\_iterator **reverse\_iterator**
- typedef \_Base::size\_type **size\_type**
- typedef std::pair< const \_Key, \_Tp > **value\_type**

#### Public Member Functions

- **multimap** (const \_Compare &\_\_comp=\_Compare(), const \_Allocator &\_\_a=\_Allocator())
- template<typename \_InputIterator, typename = std::RequireInputIter<\_InputIterator>> **multimap** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_comp=\_Compare(), const \_Allocator &\_\_a=\_Allocator())
- **multimap** (const multimap &\_\_x)
- **multimap** (const \_Base &\_\_x)
- **multimap** (multimap &&\_\_x) noexcept(is\_nothrow\_copy\_constructible< \_Compare >::value)
- **multimap** (initializer\_list< value\_type > \_\_l, const \_Compare &\_\_c=\_Compare(), const allocator\_type &\_\_a=allocator\_type())
- \_Base & **M\_base** () noexcept
- const \_Base & **M\_base** () const noexcept
- iterator **begin** () noexcept
- const\_iterator **begin** () const noexcept
- const\_iterator **cbegin** () const noexcept
- const\_iterator **end** () const noexcept
- void **clear** () noexcept
- const\_reverse\_iterator **crbegin** () const noexcept
- const\_reverse\_iterator **crend** () const noexcept
- template<typename... \_Args> iterator **emplace** (\_Args &&... \_\_args)
- template<typename... \_Args> iterator **emplace\_hint** (const\_iterator \_\_pos, \_Args &&... \_\_args)
- iterator **end** () noexcept
- const\_iterator **end** () const noexcept
- std::pair< iterator, iterator > **equal\_range** (const key\_type &\_\_x)
- std::pair< const\_iterator, const\_iterator > **equal\_range** (const key\_type &\_\_x) const
- iterator **erase** (const\_iterator \_\_position)
- iterator **erase** (iterator \_\_position)
- size\_type **erase** (const key\_type &\_\_x)
- iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- iterator **find** (const key\_type &\_\_x)
- const\_iterator **find** (const key\_type &\_\_x) const
- iterator **insert** (const value\_type &\_\_x)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type> iterator **insert** (\_Pair &&\_\_x)
- void **insert** (std::initializer\_list< value\_type > \_\_list)
- iterator **insert** (const\_iterator \_\_position, const value\_type &\_\_x)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type> iterator **insert** (const\_iterator \_\_position, \_Pair &&\_\_x)
- template<typename \_InputIterator, typename = std::RequireInputIter<\_InputIterator>> void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- iterator **lower\_bound** (const key\_type &\_\_x)
- const\_iterator **lower\_bound** (const key\_type &\_\_x) const
- multimap & **operator=** (const multimap &\_\_x)
- multimap & **operator=** (multimap &&\_\_x)
- multimap & **operator=** (initializer\_list< value\_type > \_\_l)
- reverse\_iterator **rbegin** () noexcept
- const\_reverse\_iterator **rbegin** () const noexcept

- reverse\_iterator **rend** () noexcept
- const\_reverse\_iterator **rend** () const noexcept
- void **swap** (multimap &\_\_x)
- iterator **upper\_bound** (const key\_type &\_\_x)
- const\_iterator **upper\_bound** (const key\_type &\_\_x) const

#### 4.477.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std::pair<const _Key, _Tp>>>>class std::__profile::multimap<_Key, _Tp, _Compare, _Allocator>
```

Class std::multimap wrapper with performance instrumentation.

Definition at line 41 of file profile/multimap.h.

The documentation for this class was generated from the following file:

- [profile/multimap.h](#)

#### 4.478 std::\_\_profile::multiset<\_Key, \_Compare, \_Allocator> Class Template Reference

Inherits multiset<\_Key, \_Compare, \_Allocator>.

##### Public Types

- typedef \_Allocator **allocator\_type**
- typedef \_Base::const\_iterator **const\_iterator**
- typedef \_Base::const\_pointer **const\_pointer**
- typedef \_Base::const\_reference **const\_reference**
- typedef \_Base::const\_reverse\_iterator **const\_reverse\_iterator**
- typedef \_Base::difference\_type **difference\_type**
- typedef \_Base::iterator **iterator**
- typedef \_Compare **key\_compare**
- typedef \_Key **key\_type**
- typedef \_Base::pointer **pointer**
- typedef \_Base::reference **reference**
- typedef \_Base::reverse\_iterator **reverse\_iterator**
- typedef \_Base::size\_type **size\_type**
- typedef \_Compare **value\_compare**
- typedef \_Key **value\_type**

##### Public Member Functions

- **multiset** (const \_Compare &\_\_comp=\_Compare(), const \_Allocator &\_\_a=\_Allocator())
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>> **multiset** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_comp=\_Compare(), const \_Allocator &\_\_a=\_Allocator())
- **multiset** (const multiset &\_\_x)
- **multiset** (const \_Base &\_\_x)
- **multiset** (multiset &&\_\_x) noexcept(is\_nothrow\_copy\_constructible<\_Compare>::value)
- **multiset** (initializer\_list<value\_type> \_\_l, const \_Compare &\_\_comp=\_Compare(), const allocator\_type &\_\_a=allocator\_type())

- [\\_Base](#) & [\\_M\\_base](#) () noexcept
- const [\\_Base](#) & [\\_M\\_base](#) () const noexcept
- iterator **begin** () noexcept
- const\_iterator **begin** () const noexcept
- const\_iterator **cbegin** () const noexcept
- const\_iterator **cend** () const noexcept
- void **clear** () noexcept
- const\_reverse\_iterator **crbegin** () const noexcept
- const\_reverse\_iterator **crend** () const noexcept
- template<typename... \_Args> iterator **emplace** (\_Args &&... \_\_args)
- template<typename... \_Args> iterator **emplace\_hint** (const\_iterator \_\_pos, \_Args &&... \_\_args)
- iterator **end** () noexcept
- const\_iterator **end** () const noexcept
- [std::pair](#)< iterator, iterator > **equal\_range** (const key\_type &\_\_x)
- [std::pair](#)< const\_iterator, const\_iterator > **equal\_range** (const key\_type &\_\_x) const
- iterator **erase** (const\_iterator \_\_position)
- size\_type **erase** (const key\_type &\_\_x)
- iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- iterator **find** (const key\_type &\_\_x)
- const\_iterator **find** (const key\_type &\_\_x) const
- iterator **insert** (const value\_type &\_\_x)
- iterator **insert** (value\_type &&\_\_x)
- iterator **insert** (const\_iterator \_\_position, const value\_type &\_\_x)
- iterator **insert** (const\_iterator \_\_position, value\_type &&\_\_x)
- template<typename \_InputIterator, typename = std::::RequireInputIter<\_InputIterator>> void **insert** (\_InputIterator \_\_first, \_↵  
\_InputIterator \_\_last)
- void **insert** (initializer\_list< value\_type > \_\_l)
- iterator **lower\_bound** (const key\_type &\_\_x)
- const\_iterator **lower\_bound** (const key\_type &\_\_x) const
- [multiset](#) & **operator=** (const [multiset](#) &\_\_x)
- [multiset](#) & **operator=** ([multiset](#) &&\_\_x)
- [multiset](#) & **operator=** (initializer\_list< value\_type > \_\_l)
- reverse\_iterator **rbegin** () noexcept
- const\_reverse\_iterator **rbegin** () const noexcept
- reverse\_iterator **rend** () noexcept
- const\_reverse\_iterator **rend** () const noexcept
- void **swap** ([multiset](#) &\_\_x)
- iterator **upper\_bound** (const key\_type &\_\_x)
- const\_iterator **upper\_bound** (const key\_type &\_\_x) const

#### 4.478.1 Detailed Description

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>> class std::__profile::multiset< _Key, _Compare, _Allocator >
```

Class std::multiset wrapper with performance instrumentation.

Definition at line 41 of file profile/multiset.h.

The documentation for this class was generated from the following file:

- [profile/multiset.h](#)

#### 4.479 `std::__profile::set<_Key, _Compare, _Allocator>` Class Template Reference

Inherits `set<_Key, _Compare, _Allocator>`.

##### Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `_Base::const_iterator` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `_Base::const_reverse_iterator` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `_Base::reverse_iterator` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Compare` **value\_compare**
- typedef `_Key` **value\_type**

##### Public Member Functions

- **set** (const `_Compare` &\_\_comp=\_Compare(), const `_Allocator` &\_\_a=\_Allocator())
- template<typename `_InputIterator` , typename = std::RequireInputIter<\_InputIterator>> **set** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp=\_Compare(), const `_Allocator` &\_\_a=\_Allocator())
- **set** (const `set` &\_\_x)
- **set** (const `_Base` &\_\_x)
- **set** (`set` &&\_\_x) noexcept(is\_nothrow\_copy\_constructible<\_Compare>::value)
- **set** (initializer\_list< `value_type` > \_\_l, const `_Compare` &\_\_comp=\_Compare(), const `allocator_type` &\_\_a=allocator\_type())
- `_Base` & **\_M\_base** () noexcept
- const `_Base` & **\_M\_base** () const noexcept
- iterator **begin** () noexcept
- const\_iterator **begin** () const noexcept
- const\_iterator **cbegin** () const noexcept
- const\_iterator **cend** () const noexcept
- void **clear** () noexcept
- const\_reverse\_iterator **crbegin** () const noexcept
- const\_reverse\_iterator **crend** () const noexcept
- template<typename... `_Args`> `std::pair`< iterator, bool > **emplace** (`_Args` &&... \_\_args)
- template<typename... `_Args`> iterator **emplace\_hint** (const\_iterator \_\_pos, `_Args` &&... \_\_args)
- iterator **end** () noexcept
- const\_iterator **end** () const noexcept
- `std::pair`< iterator, iterator > **equal\_range** (const `key_type` &\_\_x)
- `std::pair`< const\_iterator, const\_iterator > **equal\_range** (const `key_type` &\_\_x) const
- iterator **erase** (const\_iterator \_\_position)
- `size_type` **erase** (const `key_type` &\_\_x)
- iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)

- iterator **find** (const key\_type &\_\_x)
- const\_iterator **find** (const key\_type &\_\_x) const
- [std::pair](#)< iterator, bool > **insert** (const value\_type &\_\_x)
- [std::pair](#)< iterator, bool > **insert** (value\_type &&\_\_x)
- iterator **insert** (const\_iterator \_\_position, const value\_type &\_\_x)
- iterator **insert** (const\_iterator \_\_position, value\_type &&\_\_x)
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>> void **insert** (\_InputIterator \_\_first, \_↵  
\_InputIterator \_\_last)
- void **insert** (initializer\_list< value\_type > \_\_l)
- iterator **lower\_bound** (const key\_type &\_\_x)
- const\_iterator **lower\_bound** (const key\_type &\_\_x) const
- [set](#) & **operator=** (const [set](#) &\_\_x)
- [set](#) & **operator=** ([set](#) &&\_\_x)
- [set](#) & **operator=** (initializer\_list< value\_type > \_\_l)
- reverse\_iterator **rbegin** () noexcept
- const\_reverse\_iterator **rbegin** () const noexcept
- reverse\_iterator **rend** () noexcept
- const\_reverse\_iterator **rend** () const noexcept
- void **swap** ([set](#) &\_\_x)
- iterator **upper\_bound** (const key\_type &\_\_x)
- const\_iterator **upper\_bound** (const key\_type &\_\_x) const

#### 4.479.1 Detailed Description

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>> class std::_↵  
profile::set<_Key, _Compare, _Allocator>
```

Class std::set wrapper with performance instrumentation.

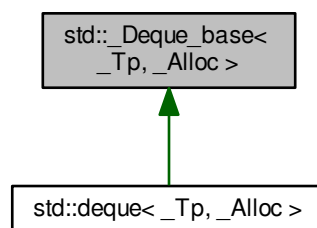
Definition at line 41 of file profile/set.h.

The documentation for this class was generated from the following file:

- [profile/set.h](#)

#### 4.480 std::\_Deque\_base<\_Tp, \_Alloc> Class Template Reference

Inheritance diagram for std::\_Deque\_base<\_Tp, \_Alloc>:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Deque_iterator`< `_Tp`, `const _Tp &`, `const _Tp *` > **const\_iterator**
- typedef `_Deque_iterator`< `_Tp`, `_Tp &`, `_Tp *` > **iterator**

## Public Member Functions

- `_Deque_base` (`size_t __num_elements`)
- `_Deque_base` (`const allocator_type &__a`, `size_t __num_elements`)
- `_Deque_base` (`const allocator_type &__a`)
- `_Deque_base` (`_Deque_base &&__x`)
- `allocator_type` **get\_allocator** () `const` `noexcept`

## Protected Types

- enum { `_S_initial_map_size` }
- typedef `_Alloc::template rebind`< `_Tp *` >::`other` **Map\_alloc\_type**
- typedef `_Alloc::template rebind`< `_Tp` >::`other` **Tp\_alloc\_type**

## Protected Member Functions

- `_Tp **` **\_M\_allocate\_map** (`size_t __n`)
- `_Tp *` **\_M\_allocate\_node** ()
- void **\_M\_create\_nodes** (`_Tp **__nstart`, `_Tp **__nfinish`)
- void **\_M\_deallocate\_map** (`_Tp **__p`, `size_t __n`)
- void **\_M\_deallocate\_node** (`_Tp *__p`)
- void **\_M\_destroy\_nodes** (`_Tp **__nstart`, `_Tp **__nfinish`)
- `_Map_alloc_type` **\_M\_get\_map\_allocator** () `const` `noexcept`
- `_Tp_alloc_type &` **\_M\_get\_Tp\_allocator** () `noexcept`
- `const _Tp_alloc_type &` **\_M\_get\_Tp\_allocator** () `const` `noexcept`
- void **\_M\_initialize\_map** (`size_t`)

## Protected Attributes

- `_Deque_impl` **\_M\_impl**

## 4.480.1 Detailed Description

```
template<typename _Tp, typename _Alloc>class std::_Deque_base< _Tp, _Alloc >
```

Deque base class. This class provides the unified face for deque's allocation. This class's constructor and destructor allocate and deallocate (but do not initialize) storage. This makes exception safety easier.

Nothing in this class ever constructs or destroys an actual `Tp` element. (Deque handles that itself.) Only/All memory management is performed here.

Definition at line 439 of file `stl_deque.h`.

## 4.480.2 Member Function Documentation

4.480.2.1 `template<typename _Tp, typename _Alloc> void std::_Deque_base<_Tp, _Alloc>::_M_initialize_map ( size_t __num_elements )` [protected]

Layout storage.



## Parameters

<code>__num_elements</code>	The count of T's for which to allocate space at first.
-----------------------------	--

## Returns

Nothing.

The initial underlying memory layout is a bit complicated...

Definition at line 582 of file `stl_deque.h`.

References `std::max()`.

The documentation for this class was generated from the following file:

- [stl\\_deque.h](#)

#### 4.481 `std::_Deque_iterator<_Tp, _Ref, _Ptr>` Struct Template Reference

## Public Types

- `typedef _Tp** _Map_pointer`
- `typedef _Deque_iterator _Self`
- `typedef _Deque_iterator<_Tp, const _Tp &, const _Tp * > const_iterator`
- `typedef ptrdiff_t difference_type`
- `typedef _Deque_iterator<_Tp, _Tp &, _Tp * > iterator`
- `typedef std::random\_access\_iterator\_tag iterator_category`
- `typedef _Ptr pointer`
- `typedef _Ref reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

## Public Member Functions

- `_Deque_iterator` (`_Tp *__x`, `_Map_pointer __y`)
- `_Deque_iterator` (`const iterator &__x`)
- `void _M_set_node` (`_Map_pointer __new_node`)
- reference `operator* () const`
- `_Self operator+` (`difference_type __n`) `const`
- `_Self & operator++ ()`
- `_Self operator++ (int)`
- `_Self & operator+=` (`difference_type __n`)
- `_Self operator-` (`difference_type __n`) `const`
- `_Self & operator-- ()`
- `_Self operator-- (int)`
- `_Self & operator-=` (`difference_type __n`)
- pointer `operator-> () const`
- reference `operator[]` (`difference_type __n`) `const`

## Static Public Member Functions

- `static size_t _S_buffer_size ()`

## Public Attributes

- `_Tp * _M_cur`
- `_Tp * _M_first`
- `_Tp * _M_last`
- `_Map_pointer _M_node`

## 4.481.1 Detailed Description

`template<typename _Tp, typename _Ref, typename _Ptr> struct std::_Deque_iterator<_Tp, _Ref, _Ptr>`

A `deque::iterator`.

Quite a bit of intelligence here. Much of the functionality of `deque` is actually passed off to this class. A `deque` holds two of these internally, marking its valid range. Access to elements is done as offsets of either of those two, relying on operator overloading in this class.

All the functions are op overloads except for `_M_set_node`.

Definition at line 106 of file `stl_deque.h`.

## 4.481.2 Member Function Documentation

4.481.2.1 `template<typename _Tp, typename _Ref, typename _Ptr> void std::_Deque_iterator<_Tp, _Ref, _Ptr>::_M_set_node( _Map_pointer __new_node ) [inline]`

Prepares to traverse `new_node`. Sets everything except `_M_cur`, which should therefore be set by the caller immediately afterwards, based on `_M_first` and `_M_last`.

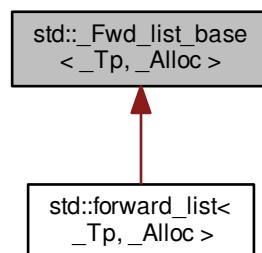
Definition at line 234 of file `stl_deque.h`.

The documentation for this struct was generated from the following file:

- [stl\\_deque.h](#)

4.482 `std::_Fwd_list_base<_Tp, _Alloc>` Struct Template Reference

Inheritance diagram for `std::_Fwd_list_base<_Tp, _Alloc>`:



## Public Types

- typedef [\\_Fwd\\_list\\_node](#)< \_Tp > **\_Node**
- typedef [\\_Fwd\\_list\\_const\\_iterator](#)< \_Tp > **const\_iterator**
- typedef [\\_Fwd\\_list\\_iterator](#)< \_Tp > **iterator**

## Public Member Functions

- **\_Fwd\_list\_base** (const \_Node\_alloc\_type &\_\_a)
- **\_Fwd\_list\_base** ( [\\_Fwd\\_list\\_base](#) && \_\_lst, const \_Node\_alloc\_type &\_\_a)
- **\_Fwd\_list\_base** ( [\\_Fwd\\_list\\_base](#) && \_\_lst)
- \_Node\_alloc\_type & **\_M\_get\_Node\_allocator** () noexcept
- const \_Node\_alloc\_type & **\_M\_get\_Node\_allocator** () const noexcept

## Protected Types

- typedef [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits](#)< \_Alloc > **\_Alloc\_traits**
- typedef [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits](#)< \_Node\_alloc\_type > **\_Node\_alloc\_traits**
- typedef \_Alloc\_traits::template rebind< [\\_Fwd\\_list\\_node](#)< \_Tp > >::other **\_Node\_alloc\_type**
- typedef \_Alloc\_traits::template rebind< \_Tp >::other **\_Tp\_alloc\_type**

## Protected Member Functions

- template<typename... \_Args> [\\_Node](#) \* **\_M\_create\_node** ( \_Args &&... \_\_args)
- [\\_Fwd\\_list\\_node\\_base](#) \* **\_M\_erase\_after** ( [\\_Fwd\\_list\\_node\\_base](#) \* \_\_pos)
- [\\_Fwd\\_list\\_node\\_base](#) \* **\_M\_erase\_after** ( [\\_Fwd\\_list\\_node\\_base](#) \* \_\_pos, [\\_Fwd\\_list\\_node\\_base](#) \* \_\_last)
- [\\_Node](#) \* **\_M\_get\_node** ()
- template<typename... \_Args> [\\_Fwd\\_list\\_node\\_base](#) \* **\_M\_insert\_after** (const\_iterator \_\_pos, \_Args &&... \_\_args)
- void **\_M\_put\_node** ( [\\_Node](#) \* \_\_p)

## Protected Attributes

- [\\_Fwd\\_list\\_impl](#) **\_M\_impl**

## 4.482.1 Detailed Description

template<typename \_Tp, typename \_Alloc> struct std::\_Fwd\_list\_base< \_Tp, \_Alloc >

Base class for forward\_list.

Definition at line 275 of file forward\_list.h.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

4.483 `std::_Fwd_list_const_iterator<_Tp>` Struct Template Reference

## Public Types

- typedef const `_Fwd_list_node<_Tp>` `_Node`
- typedef `_Fwd_list_const_iterator<_Tp>` `_Self`
- typedef ptrdiff\_t `difference_type`
- typedef `_Fwd_list_iterator<_Tp>` `iterator`
- typedef `std::forward_iterator_tag` `iterator_category`
- typedef const `_Tp` \* `pointer`
- typedef const `_Tp` & `reference`
- typedef `_Tp` `value_type`

## Public Member Functions

- `_Fwd_list_const_iterator` (const `_Fwd_list_node_base` \*\_\_n)
- `_Fwd_list_const_iterator` (const `iterator` &\_\_iter)
- `_Self` `_M_next` () const
- bool `operator!=` (const `_Self` &\_\_x) const
- reference `operator*` () const
- `_Self` & `operator++` ()
- `_Self` `operator++` (int)
- pointer `operator->` () const
- bool `operator==` (const `_Self` &\_\_x) const

## Public Attributes

- const `_Fwd_list_node_base` \* `_M_node`

## 4.483.1 Detailed Description

```
template<typename _Tp>struct std::_Fwd_list_const_iterator<_Tp>
```

A `forward_list::const_iterator`.

All the functions are op overloads.

Definition at line 188 of file `forward_list.h`.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

4.484 `std::_Fwd_list_iterator<_Tp>` Struct Template Reference

## Public Types

- typedef `_Fwd_list_node<_Tp>` `_Node`
- typedef `_Fwd_list_iterator<_Tp>` `_Self`
- typedef ptrdiff\_t `difference_type`
- typedef `std::forward_iterator_tag` `iterator_category`

- typedef `_Tp *` **pointer**
- typedef `_Tp &` **reference**
- typedef `_Tp` **value\_type**

#### Public Member Functions

- `_Fwd_list_iterator` (`_Fwd_list_node_base * __n`)
- `_Self _M_next` () const
- bool **operator!=** (const `_Self` & \_\_x) const
- reference **operator\*** () const
- `_Self` & **operator++** ()
- `_Self` **operator++** (int)
- pointer **operator->** () const
- bool **operator==** (const `_Self` & \_\_x) const

#### Public Attributes

- `_Fwd_list_node_base * _M_node`

#### 4.484.1 Detailed Description

template<typename `_Tp`>struct std::\_Fwd\_list\_iterator< `_Tp` >

A forward\_list::iterator.

All the functions are op overloads.

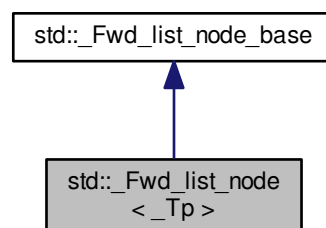
Definition at line 121 of file forward\_list.h.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

#### 4.485 std::\_Fwd\_list\_node< `_Tp` > Struct Template Reference

Inheritance diagram for std::\_Fwd\_list\_node< `_Tp` >:



## Public Member Functions

- void **\_M\_reverse\_after** () noexcept
- [\\_Fwd\\_list\\_node\\_base](#) \* **\_M\_transfer\_after** ([\\_Fwd\\_list\\_node\\_base](#) \* \_\_begin, [\\_Fwd\\_list\\_node\\_base](#) \* \_\_end)
- [\\_Tp](#) \* **\_M\_valptr** () noexcept
- const [\\_Tp](#) \* **\_M\_valptr** () const noexcept

## Public Attributes

- [\\_Fwd\\_list\\_node\\_base](#) \* **\_M\_next**
- aligned\_storage< sizeof([\\_Tp](#)), alignment\_of< [\\_Tp](#) >::value >::type **\_M\_storage**

## 4.485.1 Detailed Description

```
template<typename _Tp>struct std::_Fwd_list_node< _Tp >
```

A helper node class for forward\_list. This is just a linked list with uninitialized storage for a data value in each node. There is a sorting utility method.

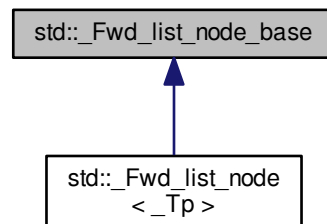
Definition at line 94 of file forward\_list.h.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

## 4.486 std::\_Fwd\_list\_node\_base Struct Reference

Inheritance diagram for std::\_Fwd\_list\_node\_base:



## Public Member Functions

- void **\_M\_reverse\_after** () noexcept
- [\\_Fwd\\_list\\_node\\_base](#) \* **\_M\_transfer\_after** ([\\_Fwd\\_list\\_node\\_base](#) \* \_\_begin, [\\_Fwd\\_list\\_node\\_base](#) \* \_\_end)

## Public Attributes

- [\\_Fwd\\_list\\_node\\_base](#) \* [\\_M\\_next](#)

## 4.486.1 Detailed Description

A helper basic node class for `forward_list`. This is just a linked list with nothing inside it. There are purely list shuffling utility methods here.

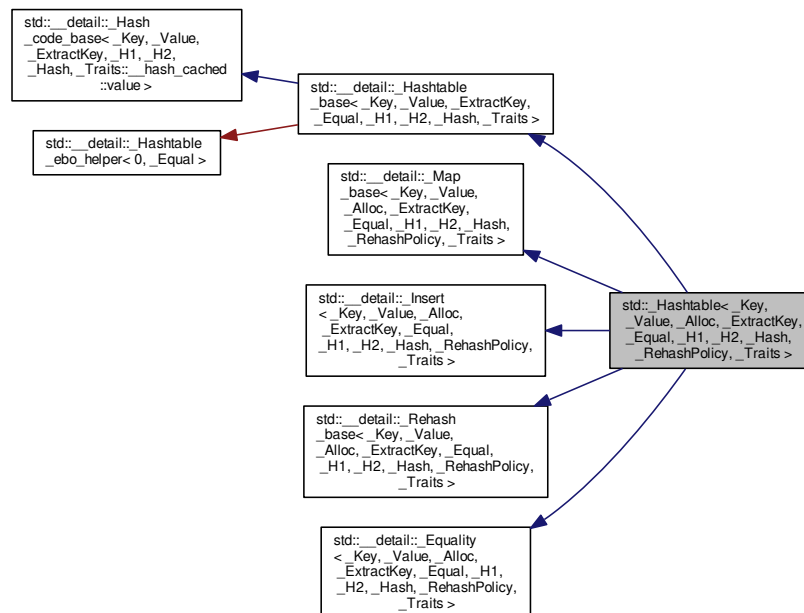
Definition at line 49 of file `forward_list.h`.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

#### 4.487 `std::_Hashtable<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >` Class Template Reference

Inheritance diagram for `std::_Hashtable<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`:



## Public Types

- typedef `_Alloc` **allocator\_type**
- using **const\_iterator** = typename `__hashtable_base::const_iterator`
- using **const\_local\_iterator** = typename `__hashtable_base::const_local_iterator`
- typedef `_Alloc::const_pointer` **const\_pointer**

- typedef `_Alloc::const_reference` **const\_reference**
- using **difference\_type** = typename `__hashtable_base::difference_type`
- using **iterator** = typename `__hashtable_base::iterator`
- typedef `_Equal` **key\_equal**
- typedef `_Key` **key\_type**
- using **local\_iterator** = typename `__hashtable_base::local_iterator`
- typedef `_Alloc::pointer` **pointer**
- typedef `_Alloc::reference` **reference**
- using **size\_type** = typename `__hashtable_base::size_type`
- typedef `_Value` **value\_type**

## Public Member Functions

- **\_Hashtable** (size\_type \_\_bucket\_hint, const \_H1 &, const \_H2 &, const \_Hash &, const \_Equal &, const \_ExtractKey &, const allocator\_type &)
- template<typename \_InputIterator > **\_Hashtable** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_bucket\_hint, const \_H1 &, const \_H2 &, const \_Hash &, const \_Equal &, const \_ExtractKey &, const allocator\_type &)
- **\_Hashtable** (const [\\_Hashtable](#) &)
- **\_Hashtable** ([\\_Hashtable](#) &&)
- **\_Hashtable** (size\_type \_\_n=10, const \_H1 & \_\_hf=\_H1(), const key\_equal & \_\_eq=key\_equal(), const allocator\_type & \_\_a=allocator\_type())
- template<typename \_InputIterator > **\_Hashtable** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n=0, const \_H1 & \_\_hf=\_H1(), const key\_equal & \_\_eq=key\_equal(), const allocator\_type & \_\_a=allocator\_type())
- **\_Hashtable** (initializer\_list< value\_type > \_\_l, size\_type \_\_n=0, const \_H1 & \_\_hf=\_H1(), const key\_equal & \_\_eq=key\_equal(), const allocator\_type & \_\_a=allocator\_type())
- const `_RehashPolicy` & **\_\_rehash\_policy** () const
- void **\_\_rehash\_policy** (const `_RehashPolicy` &)
- template<typename... \_Args> **std::pair**< typename [\\_Hashtable](#)< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits >::iterator, bool > **\_M\_emplace** (std::true\_type, \_Args &&... \_\_args)
- template<typename \_Arg > **std::pair**< typename [\\_Hashtable](#)< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits >::iterator, bool > **\_M\_insert** (\_Arg && \_\_v, std::true\_type)
- iterator **begin** () noexcept
- const\_iterator **begin** () const noexcept
- local\_iterator **begin** (size\_type \_\_n)
- const\_local\_iterator **begin** (size\_type \_\_n) const
- size\_type **bucket** (const key\_type & \_\_k) const
- size\_type **bucket\_count** () const noexcept
- size\_type **bucket\_size** (size\_type \_\_n) const
- const\_iterator **cbegin** () const noexcept
- const\_local\_iterator **cbegin** (size\_type \_\_n) const
- const\_iterator **cend** () const noexcept
- const\_local\_iterator **cend** (size\_type \_\_n) const
- void **clear** () noexcept
- size\_type **count** (const key\_type & \_\_k) const
- template<typename... \_Args> `__ireturn_type` **emplace** (\_Args &&... \_\_args)
- template<typename... \_Args> iterator **emplace\_hint** (const\_iterator, \_Args &&... \_\_args)
- bool **empty** () const noexcept
- iterator **end** () noexcept
- const\_iterator **end** () const noexcept
- local\_iterator **end** (size\_type \_\_n)
- const\_local\_iterator **end** (size\_type \_\_n) const



- `std::pair< iterator, iterator > equal_range` (const key\_type &\_\_k)
- `std::pair< const_iterator, const_iterator > equal_range` (const key\_type &\_\_k) const
- iterator `erase` (const\_iterator)
- iterator `erase` (iterator \_\_it)
- size\_type `erase` (const key\_type &\_\_k)
- iterator `erase` (const\_iterator, const\_iterator)
- iterator `find` (const key\_type &\_\_k)
- const\_iterator `find` (const key\_type &\_\_k) const
- allocator\_type `get_allocator` () const noexcept
- key\_equal `key_eq` () const
- float `load_factor` () const noexcept
- size\_type `max_bucket_count` () const noexcept
- size\_type `max_size` () const noexcept
- `_Hashtable` & `operator=` (const `_Hashtable` &\_\_ht)
- `_Hashtable` & `operator=` (`_Hashtable` &&\_\_ht)
- `_Hashtable` & `operator=` (initializer\_list< value\_type > \_\_l)
- void `rehash` (size\_type \_\_n)
- size\_type `size` () const noexcept
- void `swap` (`_Hashtable` &)

#### Protected Member Functions

- size\_type `_M_bucket_index` (\_\_node\_type \*\_\_n) const
- size\_type `_M_bucket_index` (const key\_type &\_\_k, \_\_hash\_code \_\_c) const
- template<typename... \_Args> `std::pair< iterator, bool > _M_emplace` (std::true\_type, \_Args &&... \_\_args)
- template<typename... \_Args> iterator `_M_emplace` (std::false\_type, \_Args &&... \_\_args)
- const\_Equal & `_M_eq` () const
- \_Equal & `_M_eq` ()
- bool `_M_equals` (const\_Key &\_\_k, \_\_hash\_code \_\_c, \_\_node\_type \*\_\_n) const
- size\_type `_M_erase` (std::true\_type, const key\_type &)
- size\_type `_M_erase` (std::false\_type, const key\_type &)
- iterator `_M_erase` (size\_type \_\_bkt, \_\_node\_base \*\_\_prev\_n, \_\_node\_type \*\_\_n)
- \_\_node\_base \* `_M_find_before_node` (size\_type, const key\_type &, \_\_hash\_code) const
- \_\_node\_type \* `_M_find_node` (size\_type \_\_bkt, const key\_type &\_\_key, \_\_hash\_code \_\_c) const
- \_\_node\_base \* `_M_get_previous_node` (size\_type \_\_bkt, \_\_node\_base \*\_\_n)
- template<typename \_Arg > `std::pair< iterator, bool > _M_insert` (\_Arg &&, std::true\_type)
- template<typename \_Arg > iterator `_M_insert` (\_Arg &&, std::false\_type)
- void `_M_insert_bucket_begin` (size\_type, \_\_node\_type \*)
- iterator `_M_insert_multi_node` (\_\_hash\_code \_\_code, \_\_node\_type \*\_\_n)
- iterator `_M_insert_unique_node` (size\_type \_\_bkt, \_\_hash\_code \_\_code, \_\_node\_type \*\_\_n)
- void `_M_remove_bucket_begin` (size\_type \_\_bkt, \_\_node\_type \*\_\_next\_n, size\_type \_\_next\_bkt)
- void `_M_swap` (\_Hashtable\_base &\_\_x)

#### Friends

- template<typename \_Keya, typename \_Valuea, typename \_Alloca, typename \_ExtractKeya, typename \_Equala, typename \_H1a, typename \_H2a, typename \_Hasha, typename \_RehashPolicya, typename \_Traitsa, bool \_Constant\_iteratorsa, bool \_Unique\_keysa> struct `__detail::Insert`
- template<typename \_Keya, typename \_Valuea, typename \_Alloca, typename \_ExtractKeya, typename \_Equala, typename \_H1a, typename \_H2a, typename \_Hasha, typename \_RehashPolicya, typename \_Traitsa > struct `__detail::Insert_base`
- template<typename \_Keya, typename \_Valuea, typename \_Alloca, typename \_ExtractKeya, typename \_Equala, typename \_H1a, typename \_H2a, typename \_Hasha, typename \_RehashPolicya, typename \_Traitsa, bool \_Unique\_keysa> struct `__detail::Map_base`

#### 4.487.1 Detailed Description

`template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2, typename _Hash, typename _RehashPolicy, typename _Traits> class std::_Hashtable<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`

Primary class template `_Hashtable`.

##### Template Parameters

<code>_Value</code>	CopyConstructible type.
<code>_Key</code>	CopyConstructible type.
<code>_Alloc</code>	An allocator type ([lib.allocator.requirements]) whose <code>_Alloc::value_type</code> is <code>_Value</code> . As a conforming extension, we allow for <code>_Alloc::value_type != _Value</code> .
<code>_ExtractKey</code>	Function object that takes an object of type <code>_Value</code> and returns a value of type <code>_Key</code> .
<code>_Equal</code>	Function object that takes two objects of type <code>k</code> and returns a bool-like value that is true if the two objects are considered equal.
<code>_H1</code>	The hash function. A unary function object with argument type <code>_Key</code> and result type <code>size_t</code> . Return values should be distributed over the entire range <code>[0, numeric_limits&lt;size_t&gt;:max())</code> .
<code>_H2</code>	The range-hashing function (in the terminology of Tavori and Dreizin). A binary function object whose argument types and result type are all <code>size_t</code> . Given arguments <code>r</code> and <code>N</code> , the return value is in the range <code>[0, N)</code> .
<code>_Hash</code>	The ranged hash function (Tavori and Dreizin). A binary function whose argument types are <code>_Key</code> and <code>size_t</code> and whose result type is <code>size_t</code> . Given arguments <code>k</code> and <code>N</code> , the return value is in the range <code>[0, N)</code> . Default: <code>hash(k, N) = h2(h1(k), N)</code> . If <code>_Hash</code> is anything other than the default, <code>_H1</code> and <code>_H2</code> are ignored.
<code>_RehashPolicy</code>	Policy class with three members, all of which govern the bucket count. <code>_M_next_bkt(n)</code> returns a bucket count no smaller than <code>n</code> . <code>_M_bkt_for_elements(n)</code> returns a bucket count appropriate for an element count of <code>n</code> . <code>_M_need_rehash(n_bkt, n_elt, n_ins)</code> determines whether, if the current bucket count is <code>n_bkt</code> and the current element count is <code>n_elt</code> , we need to increase the bucket count. If so, returns <code>make_pair(true, n)</code> , where <code>n</code> is the new bucket count. If not, returns <code>make_pair(false, &lt;anything&gt;)</code> .
<code>_Traits</code>	Compile-time class with three boolean <code>std::integral_constant</code> members: <code>__cache_hash_code</code> , <code>__constant_iterators</code> , <code>__unique_keys</code> .

Each `_Hashtable` data structure has:

- `_Bucket[] _M_buckets`
- `_Hash_node_base _M_bbegin`
- `size_type _M_bucket_count`
- `size_type _M_element_count`

with `_Bucket` being `_Hash_node*` and `_Hash_node` containing:

- `_Hash_node* _M_next`
- `Tp _M_value`
- `size_t _M_hash_code` if `cache_hash_code` is true

In terms of Standard containers the hashtable is like the aggregation of:

- `std::forward_list<_Node>` containing the elements
- `std::vector<std::forward_list<_Node>::iterator>` representing the buckets

The non-empty buckets contain the node before the first node in the bucket. This design makes it possible to implement something like `std::forward_list::insert_after` on container insertion and `std::forward_list::erase_after` on container erase calls. `_M_before_begin` is equivalent to `std::forward_list::before_begin`. Empty buckets contain `nullptr`. Note that one of the non-empty buckets contains `&_M_before_begin` which is not a dereferenceable node so the node pointer in a bucket shall never be dereferenced, only its next node can be.

Walking through a bucket's nodes requires a check on the hash code to see if each node is still in the bucket. Such a design assumes a quite efficient hash functor and is one of the reasons it is highly advisable to set `__cache_hash_code` to `true`.

The container iterators are simply built from nodes. This way incrementing the iterator is perfectly efficient independent of how many empty buckets there are in the container.

On insert we compute the element's hash code and use it to find the bucket index. If the element must be inserted in an empty bucket we add it at the beginning of the singly linked list and make the bucket point to `_M_before_begin`. The bucket that used to point to `_M_before_begin`, if any, is updated to point to its new before begin node.

On erase, the simple iterator design requires using the hash functor to get the index of the bucket to update. For this reason, when `__cache_hash_code` is set to `false` the hash functor must not throw and this is enforced by a static assertion.

Functionality is implemented by decomposition into base classes, where the derived `_Hashtable` class is used in `_Map_base`, `_Insert`, `_Rehash_base`, and `_Equality` base classes to access the "this" pointer. `_Hashtable_base` is used in the base classes as a non-recursive, fully-completed-type so that detailed nested type information, such as iterator type and node type, can be used. This is similar to the "Curiously Recurring Template Pattern" (CRTP) technique, but uses a reconstructed, not explicitly passed, template pattern.

Base class templates are:

- `__detail::_Hashtable_base`
- `__detail::_Map_base`
- `__detail::_Insert`
- `__detail::_Rehash_base`
- `__detail::_Equality`

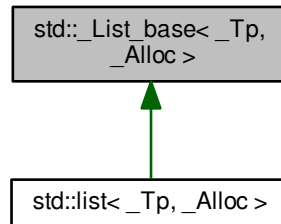
Definition at line 174 of file `bits/hashtable.h`.

The documentation for this class was generated from the following file:

- [bits/hashtable.h](#)

## 4.488 std::\_List\_base&lt; \_Tp, \_Alloc &gt; Class Template Reference

Inheritance diagram for std::\_List\_base< \_Tp, \_Alloc >:



## Public Types

- typedef `_Alloc` **allocator\_type**

## Public Member Functions

- **\_List\_base** (const `_Node_alloc_type` &\_\_a)
- **\_List\_base** (`_List_base` &&\_\_x)
- void **\_M\_clear** ()
- `_Node_alloc_type` & **\_M\_get\_Node\_allocator** () noexcept
- const `_Node_alloc_type` & **\_M\_get\_Node\_allocator** () const noexcept
- `_Tp_alloc_type` **\_M\_get\_Tp\_allocator** () const noexcept
- void **\_M\_init** ()
- `allocator_type` **get\_allocator** () const noexcept

## Protected Types

- typedef `_Alloc::template rebind< _List_node< _Tp > >::other` **\_Node\_alloc\_type**
- typedef `_Alloc::template rebind< _Tp >::other` **\_Tp\_alloc\_type**

## Protected Member Functions

- `_List_node`< \_Tp > \* **\_M\_get\_node** ()
- void **\_M\_put\_node** (`_List_node`< \_Tp > \*\_\_p)

## Protected Attributes

- `_List_impl` **\_M\_impl**

## 4.488.1 Detailed Description

```
template<typename _Tp, typename _Alloc>class std::_List_base< _Tp, _Alloc >
```

See `bits/stl_deque.h`'s `_Deque_base` for an explanation.

Definition at line 289 of file `stl_list.h`.

The documentation for this class was generated from the following file:

- [stl\\_list.h](#)

4.489 `std::_List_const_iterator< _Tp >` Struct Template Reference

## Public Types

- typedef const `_List_node`< \_Tp > `_Node`
- typedef `_List_const_iterator`< \_Tp > `_Self`
- typedef ptrdiff\_t `difference_type`
- typedef `_List_iterator`< \_Tp > `iterator`
- typedef `std::bidirectional_iterator_tag` `iterator_category`
- typedef const \_Tp \* `pointer`
- typedef const \_Tp & `reference`
- typedef \_Tp `value_type`

## Public Member Functions

- `_List_const_iterator` (const `__detail::_List_node_base` \* \_\_x)
- `_List_const_iterator` (const `iterator` & \_\_x)
- bool `operator!=` (const `_Self` & \_\_x) const
- reference `operator*` () const
- `_Self` & `operator++` ()
- `_Self` `operator++` (int)
- `_Self` & `operator--` ()
- `_Self` `operator--` (int)
- pointer `operator->` () const
- bool `operator==` (const `_Self` & \_\_x) const

## Public Attributes

- const `__detail::_List_node_base` \* `_M_node`

## 4.489.1 Detailed Description

```
template<typename _Tp>struct std::_List_const_iterator< _Tp >
```

A `list::const_iterator`.

All the functions are op overloads.

Definition at line 200 of file `stl_list.h`.

The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

## 4.490 `std::_List_iterator< _Tp >` Struct Template Reference

### Public Types

- typedef `_List_node< _Tp > _Node`
- typedef `_List_iterator< _Tp > _Self`
- typedef `ptrdiff_t difference_type`
- typedef `std::bidirectional_iterator_tag iterator_category`
- typedef `_Tp * pointer`
- typedef `_Tp & reference`
- typedef `_Tp value_type`

### Public Member Functions

- `_List_iterator` (`__detail::_List_node_base * __x`)
- `bool operator!=` (`const _Self & __x`) `const`
- `reference operator*` () `const`
- `_Self & operator++` ()
- `_Self operator++` (`int`)
- `_Self & operator--` ()
- `_Self operator--` (`int`)
- `pointer operator->` () `const`
- `bool operator==` (`const _Self & __x`) `const`

### Public Attributes

- `__detail::_List_node_base * _M_node`

#### 4.490.1 Detailed Description

```
template<typename _Tp>struct std::_List_iterator< _Tp >
```

A list::iterator.

All the functions are op overloads.

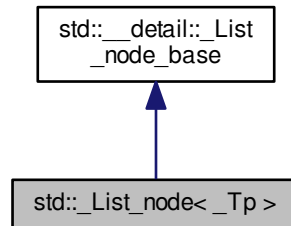
Definition at line 125 of file `stl_list.h`.

The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

#### 4.491 `std::_List_node<_Tp>` Struct Template Reference

Inheritance diagram for `std::_List_node<_Tp>`:



##### Public Member Functions

- `template<typename... _Args> _List_node (_Args &&... __args)`
- `void _M_hook (_List_node_base *const __position) noexcept`
- `void _M_reverse () noexcept`
- `void _M_transfer (_List_node_base *const __first, _List_node_base *const __last) noexcept`
- `void _M_unhook () noexcept`

##### Static Public Member Functions

- `static void swap (_List_node_base &__x, _List_node_base &__y) noexcept`

##### Public Attributes

- `_Tp _M_data`
- `_List_node_base * _M_next`
- `_List_node_base * _M_prev`

##### 4.491.1 Detailed Description

`template<typename _Tp> struct std::_List_node<_Tp>`

An actual node in the list.

Definition at line 106 of file `stl_list.h`.

##### 4.491.2 Member Data Documentation

###### 4.491.2.1 `template<typename _Tp> _Tp std::_List_node<_Tp>::_M_data`

< User's data.

Definition at line 109 of file `stl_list.h`.

The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

## 4.492 `std::_Temporary_buffer<_ForwardIterator, _Tp>` Class Template Reference

### Public Types

- typedef pointer **iterator**
- typedef value\_type \* **pointer**
- typedef ptrdiff\_t **size\_type**
- typedef \_Tp **value\_type**

### Public Member Functions

- [\\_Temporary\\_buffer](#) (`_ForwardIterator __first, _ForwardIterator __last`)
- iterator [begin](#) ()
- iterator [end](#) ()
- size\_type [requested\\_size](#) () const
- size\_type [size](#) () const

### Protected Attributes

- pointer **\_M\_buffer**
- size\_type **\_M\_len**
- size\_type **\_M\_original\_len**

#### 4.492.1 Detailed Description

```
template<typename _ForwardIterator, typename _Tp> class std::_Temporary_buffer<_ForwardIterator, _Tp>
```

This class is used in two places: `stl_algo.h` and `ext/memory`, where it is wrapped as the `temporary_buffer` class. See `temporary_buffer` docs for more notes.

Definition at line 122 of file `stl_tempbuf.h`.

#### 4.492.2 Constructor & Destructor Documentation

```
4.492.2.1 template<typename _ForwardIterator, typename _Tp> std::_Temporary_buffer<_ForwardIterator, _Tp>
::_Temporary_buffer ( _ForwardIterator __first, _ForwardIterator __last )
```

Constructs a temporary buffer of a size somewhere between zero and the size of the given range.

Definition at line 244 of file `stl_tempbuf.h`.

References `std::pair<_T1, _T2>::first`, `std::get_temporary_buffer()`, `std::return_temporary_buffer()`, and `std::pair<_T1, _T2>::second`.



#### 4.492.3 Member Function Documentation

4.492.3.1 `template<typename _ForwardIterator, typename _Tp> iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::begin ( ) [inline]`

As per Table mumble.

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

Referenced by `std::inplace_merge()`, `std::stable_partition()`, and `std::stable_sort()`.

4.492.3.2 `template<typename _ForwardIterator, typename _Tp> iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::end ( ) [inline]`

As per Table mumble.

Definition at line 156 of file `stl_tempbuf.h`.

4.492.3.3 `template<typename _ForwardIterator, typename _Tp> size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::requested_size ( ) const [inline]`

Returns the size requested by the constructor; may be `>size()`.

Definition at line 146 of file `stl_tempbuf.h`.

Referenced by `std::stable_partition()`.

4.492.3.4 `template<typename _ForwardIterator, typename _Tp> size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::size ( ) const [inline]`

As per Table mumble.

Definition at line 141 of file `stl_tempbuf.h`.

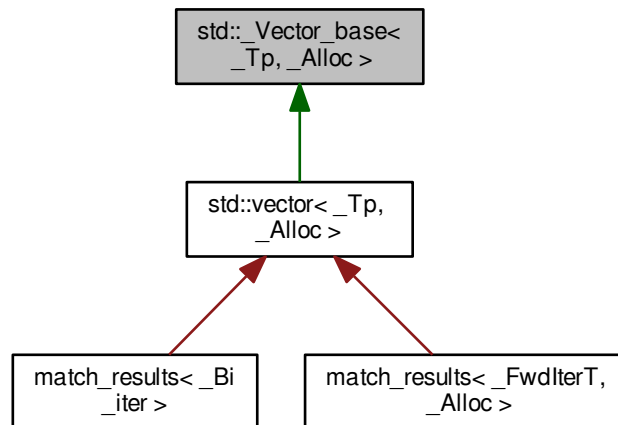
Referenced by `std::inplace_merge()`, `std::stable_partition()`, and `std::stable_sort()`.

The documentation for this class was generated from the following file:

- [stl\\_tempbuf.h](#)

## 4.493 std::\_Vector\_base&lt; \_Tp, \_Alloc &gt; Struct Template Reference

Inheritance diagram for std::\_Vector\_base< \_Tp, \_Alloc >:



## Public Types

- typedef `__gnu_cxx::__alloc_traits< _Alloc >::template rebind< _Tp >::other _Tp_alloc_type`
- typedef `_Alloc allocator_type`
- typedef `__gnu_cxx::__alloc_traits< _Tp_alloc_type >::pointer pointer`

## Public Member Functions

- `_Vector_base` (const allocator\_type &\_\_a)
- `_Vector_base` (size\_t \_\_n)
- `_Vector_base` (size\_t \_\_n, const allocator\_type &\_\_a)
- `_Vector_base` (\_Tp\_alloc\_type &&\_\_a)
- `_Vector_base` (\_Vector\_base &&\_\_x)
- `_Vector_base` (\_Vector\_base &&\_\_x, const allocator\_type &\_\_a)
- pointer `_M_allocate` (size\_t \_\_n)
- void `_M_deallocate` (pointer \_\_p, size\_t \_\_n)
- `_Tp_alloc_type & _M_get_Tp_allocator` () noexcept
- const `_Tp_alloc_type & _M_get_Tp_allocator` () const noexcept
- allocator\_type `get_allocator` () const noexcept

## Public Attributes

- `_Vector_impl _M_impl`

## 4.493.1 Detailed Description

```
template<typename _Tp, typename _Alloc> struct std::_Vector_base< _Tp, _Alloc >
```

See `bits/stl_deque.h`'s `_Deque_base` for an explanation.

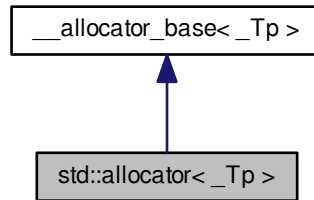
Definition at line 72 of file `stl_vector.h`.

The documentation for this struct was generated from the following file:

- [stl\\_vector.h](#)

4.494 `std::allocator< _Tp >` Class Template Reference

Inheritance diagram for `std::allocator< _Tp >`:



## Public Types

- typedef const `_Tp` \* **const\_pointer**
- typedef const `_Tp` & **const\_reference**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_Tp` \* **pointer**
- typedef `true_type` **propagate\_on\_container\_move\_assignment**
- typedef `_Tp` & **reference**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- **allocator** (const [allocator](#) &\_\_a) throw ()
- template<typename `_Tp1` > **allocator** (const [allocator](#)< `_Tp1` > &) throw ()
- pointer **address** (reference \_\_x) const noexcept
- const\_pointer **address** (const\_reference \_\_x) const noexcept
- pointer **allocate** (size\_type \_\_n, const void \*==0)
- template<typename `_Up`, typename... `_Args`> void **construct** (`_Up` \*\_\_p, `_Args` &&...\_\_args)
- void **deallocate** (pointer \_\_p, size\_type)
- template<typename `_Up` > void **destroy** (`_Up` \*\_\_p)
- size\_type **max\_size** () const noexcept

## 4.494.1 Detailed Description

```
template<typename _Tp>class std::allocator< _Tp >
```

The *standard* allocator, as per [20.4].

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt04ch11.html> for further details.

## Template Parameters

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

Definition at line 92 of file `allocator.h`.

The documentation for this class was generated from the following file:

- [allocator.h](#)

4.495 `std::allocator< void >` Class Template Reference

## Public Types

- typedef const void \* **const\_pointer**
- typedef ptrdiff\_t **difference\_type**
- typedef void \* **pointer**
- typedef true\_type **propagate\_on\_container\_move\_assignment**
- typedef size\_t **size\_type**
- typedef void **value\_type**

## 4.495.1 Detailed Description

```
template<>class std::allocator< void >
```

`allocator<void>` specialization.

Definition at line 63 of file `allocator.h`.

The documentation for this class was generated from the following file:

- [allocator.h](#)

4.496 `std::allocator_arg_t` Struct Reference

## 4.496.1 Detailed Description

[`allocator.tag`]

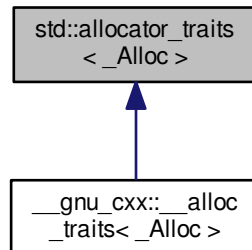
Definition at line 39 of file `uses_allocator.h`.

The documentation for this struct was generated from the following file:

- `uses_allocator.h`

#### 4.497 `std::allocator_traits<_Alloc>` Struct Template Reference

Inheritance diagram for `std::allocator_traits<_Alloc>`:



##### Public Types

- typedef `_Alloc` [allocator\\_type](#)
- typedef `__const_pointer` [const\\_pointer](#)
- typedef `__const_void_pointer` [const\\_void\\_pointer](#)
- typedef `__difference_type` [difference\\_type](#)
- typedef `__pointer` [pointer](#)
- typedef `__propagate_on_container_copy_assignment` [propagate\\_on\\_container\\_copy\\_assignment](#)
- typedef `__propagate_on_container_move_assignment` [propagate\\_on\\_container\\_move\\_assignment](#)
- typedef `__propagate_on_container_swap` [propagate\\_on\\_container\\_swap](#)
- template<typename `_Tp`> using **rebind\_alloc** = typename `__alloc_traits::rebind_alloc<_Alloc, _Tp>`::\_\_type
- template<typename `_Tp`> using **rebind\_traits** = [allocator\\_traits](#)< `rebind_alloc<_Alloc, _Tp>`>
- typedef `__size_type` [size\\_type](#)
- typedef `_Alloc::value_type` [value\\_type](#)
- typedef `__void_pointer` [void\\_pointer](#)

##### Static Public Member Functions

- static [pointer allocate](#) (`_Alloc &__a`, [size\\_type \\_\\_n](#))
- static [pointer allocate](#) (`_Alloc &__a`, [size\\_type \\_\\_n](#), [const\\_void\\_pointer \\_\\_hint](#))
- template<typename `_Tp`, typename... `_Args`> static auto [construct](#) (`_Alloc &__a`, `_Tp *__p`, `_Args &&... __args`) -> `decltype(_S_construct(__a, __p, std::forward<_Args>(__args)...))`
- static void [deallocate](#) (`_Alloc &__a`, [pointer \\_\\_p](#), [size\\_type \\_\\_n](#))
- template<class `_Tp`> static void [destroy](#) (`_Alloc &__a`, `_Tp *__p`)
- static [size\\_type max\\_size](#) (`const _Alloc &__a`)
- static `_Alloc` [select\\_on\\_container\\_copy\\_construction](#) (`const _Alloc &__rhs`)

#### 4.497.1 Detailed Description

```
template<typename _Alloc> struct std::allocator_traits<_Alloc>
```

Uniform interface to all allocator types.

Definition at line 85 of file bits/alloc\_traits.h.

#### 4.497.2 Member Typedef Documentation

**4.497.2.1** `template<typename _Alloc> typedef _Alloc std::allocator_traits<_Alloc>::allocator_type`

The allocator type.

Definition at line 88 of file bits/alloc\_traits.h.

**4.497.2.2** `template<typename _Alloc> typedef __const_pointer std::allocator_traits<_Alloc>::const_pointer`

The allocator's const pointer type.

`Alloc::const_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const value_type>`

Definition at line 118 of file bits/alloc\_traits.h.

**4.497.2.3** `template<typename _Alloc> typedef __const_void_pointer std::allocator_traits<_Alloc>::const_void_pointer`

The allocator's const void pointer type.

`Alloc::const_void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const void>`

Definition at line 140 of file bits/alloc\_traits.h.

**4.497.2.4** `template<typename _Alloc> typedef __difference_type std::allocator_traits<_Alloc>::difference_type`

The allocator's difference type.

`Alloc::difference_type` if that type exists, otherwise `pointer_traits<pointer>::difference_type`

Definition at line 151 of file bits/alloc\_traits.h.

**4.497.2.5** `template<typename _Alloc> typedef __pointer std::allocator_traits<_Alloc>::pointer`

The allocator's pointer type.

`Alloc::pointer` if that type exists, otherwise `value_type*`

Definition at line 107 of file bits/alloc\_traits.h.

**4.497.2.6** `template<typename _Alloc> typedef __propagate_on_container_copy_assignment std::allocator_traits<_Alloc>::propagate_on_container_copy_assignment`

How the allocator is propagated on copy assignment.

`Alloc::propagate_on_container_copy_assignment` if that type exists, otherwise `false_type`

Definition at line 174 of file bits/alloc\_traits.h.

**4.497.2.7** `template<typename _Alloc> typedef __propagate_on_container_move_assignment std::allocator_traits< _Alloc >::propagate_on_container_move_assignment`

How the allocator is propagated on move assignment.

`Alloc::propagate_on_container_move_assignment` if that type exists, otherwise `false_type`

Definition at line 186 of file `bits/alloc_traits.h`.

**4.497.2.8** `template<typename _Alloc> typedef __propagate_on_container_swap std::allocator_traits< _Alloc >::propagate_on_container_swap`

How the allocator is propagated on swap.

`Alloc::propagate_on_container_swap` if that type exists, otherwise `false_type`

Definition at line 197 of file `bits/alloc_traits.h`.

**4.497.2.9** `template<typename _Alloc> typedef __size_type std::allocator_traits< _Alloc >::size_type`

The allocator's size type.

`Alloc::size_type` if that type exists, otherwise `make_unsigned<difference_type>::type`

Definition at line 162 of file `bits/alloc_traits.h`.

**4.497.2.10** `template<typename _Alloc> typedef _Alloc::value_type std::allocator_traits< _Alloc >::value_type`

The allocated type.

Definition at line 90 of file `bits/alloc_traits.h`.

**4.497.2.11** `template<typename _Alloc> typedef __void_pointer std::allocator_traits< _Alloc >::void_pointer`

The allocator's void pointer type.

`Alloc::void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<void>`

Definition at line 129 of file `bits/alloc_traits.h`.

#### 4.497.3 Member Function Documentation

**4.497.3.1** `template<typename _Alloc> static pointer std::allocator_traits< _Alloc >::allocate ( _Alloc & __a, size_type __n ) [inline],[static]`

Allocate memory.

Parameters

<code>__a</code>	An allocator.
<code>__n</code>	The number of objects to allocate space for.

Calls `a.allocate(n)`

Definition at line 350 of file `bits/alloc_traits.h`.

**4.497.3.2** `template<typename _Alloc> static pointer std::allocator_traits< _Alloc >::allocate ( _Alloc & __a, size_type __n, const_void_pointer __hint ) [inline],[static]`

Allocate memory.

## Parameters

<code>__a</code>	An allocator.
<code>__n</code>	The number of objects to allocate space for.
<code>__hint</code>	Aid to locality.

## Returns

Memory of suitable size and alignment for  $n$  objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

Definition at line 365 of file `bits/alloc_traits.h`.

```
4.497.3.3 template<typename _Alloc> template<typename _Tp, typename... _Args> static auto std::allocator_traits<
_Alloc>::construct ( _Alloc & __a, _Tp * __p, _Args &&... __args ) -> decltype(_S_construct(__a, __p,
std::forward<_Args>(__args)...)) [inline], [static]
```

Construct an object of type `_Tp`.

## Parameters

<code>__a</code>	An allocator.
<code>__p</code>	Pointer to memory of suitable size and alignment for <code>Tp</code>
<code>__args</code>	Constructor arguments.

Calls `__a.construct(__p, std::forward<Args>(__args)...)`  if that expression is well-formed, otherwise uses placement-new to construct an object of type `_Tp` at location `__p` from the arguments `__args...`

Definition at line 391 of file `bits/alloc_traits.h`.

```
4.497.3.4 template<typename _Alloc> static void std::allocator_traits<_Alloc>::deallocate ( _Alloc & __a, pointer __p,
size_type __n ) [inline], [static]
```

Deallocate memory.

## Parameters

<code>__a</code>	An allocator.
<code>__p</code>	Pointer to the memory to deallocate.
<code>__n</code>	The number of objects space was allocated for.

Calls `a.deallocate(p, n)`

Definition at line 376 of file `bits/alloc_traits.h`.

```
4.497.3.5 template<typename _Alloc> template<class _Tp> static void std::allocator_traits<_Alloc>::destroy ( _Alloc &
__a, _Tp * __p ) [inline], [static]
```

Destroy an object of type `_Tp`.

## Parameters

<code>__a</code>	An allocator.
<code>__p</code>	Pointer to the object to destroy

Calls `__a.destroy(__p)` if that expression is well-formed, otherwise calls `__p->~_Tp()`

Definition at line 404 of file `bits/alloc_traits.h`.



4.497.3.6 `template<typename _Alloc> static size_type std::allocator_traits<_Alloc>::max_size ( const _Alloc & __a )`  
`[inline], [static]`

The maximum supported allocation size.

## Parameters

<code>__a</code>	An allocator.
------------------	---------------

## Returns

`__a.max_size()` or `numeric_limits<size_type>::max()`

Returns `__a.max_size()` if that expression is well-formed, otherwise returns `numeric_limits<size_type>::max()`

Definition at line 415 of file `bits/alloc_traits.h`.

Referenced by `std::forward_list<_Tp, _Alloc>::max_size()`.

4.497.3.7 `template<typename _Alloc> static _Alloc std::allocator_traits<_Alloc>::select_on_container_copy_construction (const _Alloc &__rhs) [inline], [static]`

Obtain an allocator to use when copying a container.

## Parameters

<code>__rhs</code>	An allocator.
--------------------	---------------

## Returns

`__rhs.select_on_container_copy_construction()` or `__rhs`

Returns `__rhs.select_on_container_copy_construction()` if that expression is well-formed, otherwise returns `__rhs`

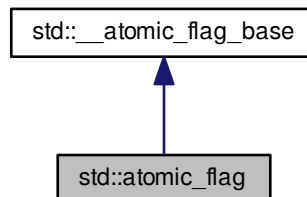
Definition at line 427 of file `bits/alloc_traits.h`.

The documentation for this struct was generated from the following file:

- [bits/alloc\\_traits.h](#)

## 4.498 std::atomic\_flag Struct Reference

Inheritance diagram for `std::atomic_flag`:



## Public Member Functions

- **atomic\_flag** (const [atomic\\_flag](#) &)=delete
- constexpr **atomic\_flag** (bool \_\_i) noexcept
- void **clear** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- void **clear** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatilenoexcept
- **atomic\_flag** & **operator=** (const [atomic\\_flag](#) &)=delete
- **atomic\_flag** & **operator=** (const [atomic\\_flag](#) &) volatile=delete
- bool **test\_and\_set** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **test\_and\_set** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatilenoexcept

## Public Attributes

- [\\_\\_atomic\\_flag\\_data\\_type](#) **\_M\_i**

## 4.498.1 Detailed Description

[atomic\\_flag](#)

Definition at line 271 of file [atomic\\_base.h](#).

The documentation for this struct was generated from the following file:

- [atomic\\_base.h](#)

4.499 [std::auto\\_ptr<\\_Tp>](#) Class Template Reference

## Public Types

- typedef [\\_Tp](#) [element\\_type](#)

## Public Member Functions

- [auto\\_ptr](#) ([element\\_type](#) \*\_\_p=0) throw ()
- [auto\\_ptr](#) ([auto\\_ptr](#) &\_\_a) throw ()
- template<typename [\\_Tp1](#) > [auto\\_ptr](#) ([auto\\_ptr](#)< [\\_Tp1](#) > &\_\_a) throw ()
- [auto\\_ptr](#) ([auto\\_ptr\\_ref](#)< [element\\_type](#) > \_\_ref) throw ()
- ~[auto\\_ptr](#) ()
- [element\\_type](#) \* [get](#) () const throw ()
- template<typename [\\_Tp1](#) > **operator** [auto\\_ptr](#)< [\\_Tp1](#) > () throw ()
- template<typename [\\_Tp1](#) > **operator** [auto\\_ptr\\_ref](#)< [\\_Tp1](#) > () throw ()
- [element\\_type](#) & **operator\*** () const throw ()
- [element\\_type](#) \* **operator->** () const throw ()
- [auto\\_ptr](#) & **operator=** ([auto\\_ptr](#) &\_\_a) throw ()
- template<typename [\\_Tp1](#) > [auto\\_ptr](#) & **operator=** ([auto\\_ptr](#)< [\\_Tp1](#) > &\_\_a) throw ()
- [auto\\_ptr](#) & **operator=** ([auto\\_ptr\\_ref](#)< [element\\_type](#) > \_\_ref) throw ()
- [element\\_type](#) \* [release](#) () throw ()
- void [reset](#) ([element\\_type](#) \*\_\_p=0) throw ()

## 4.499.1 Detailed Description

```
template<typename _Tp>class std::auto_ptr<_Tp>
```

A simple smart pointer providing strict ownership semantics.

The Standard says:

An `auto_ptr` owns the object it holds a pointer to. Copying an `auto_ptr` copies the pointer and transfers ownership to the destination. If more than one `auto_ptr` owns the same object at the same time the behavior of the program is undefined.

The uses of `auto_ptr` include providing temporary exception-safety for dynamically allocated memory, passing ownership of dynamically allocated memory to a function, and returning dynamically allocated memory from a function. `auto_ptr` does not meet the CopyConstructible requirements for Standard Library `container` elements and thus instantiating a Standard Library container with an `auto_ptr` results in undefined behavior.

Quoted from [20.4.5]/3.

Good examples of what can and cannot be done with `auto_ptr` can be found in the libstdc++ testsuite.

\_GLIBCXX\_RESOLVE\_LIB\_DEFECTS 127. `auto_ptr<>` conversion issues These resolutions have all been incorporated.

Definition at line 87 of file `auto_ptr.h`.

## 4.499.2 Member Typedef Documentation

4.499.2.1 `template<typename _Tp> typedef _Tp std::auto_ptr<_Tp>::element_type`

The pointed-to type.

Definition at line 94 of file `auto_ptr.h`.

## 4.499.3 Constructor &amp; Destructor Documentation

4.499.3.1 `template<typename _Tp> std::auto_ptr<_Tp>::auto_ptr ( element_type * __p = 0 ) throw () [inline], [explicit]`

An `auto_ptr` is usually constructed from a raw pointer.

Parameters

<code>__p</code>	A pointer (defaults to NULL).
------------------	-------------------------------

This object now *owns* the object pointed to by `__p`.

Definition at line 103 of file `auto_ptr.h`.

4.499.3.2 `template<typename _Tp> std::auto_ptr<_Tp>::auto_ptr ( auto_ptr<_Tp> & __a ) throw () [inline]`

An `auto_ptr` can be constructed from another `auto_ptr`.

**Parameters**

<code>__a</code>	Another <code>auto_ptr</code> of the same type.
------------------	---

This object now *owns* the object previously owned by `__a`, which has given up ownership.

Definition at line 112 of file `auto_ptr.h`.

**4.499.3.3** `template<typename _Tp> template<typename _Tp1 > std::auto_ptr<_Tp>::auto_ptr ( auto_ptr<_Tp1 > & __a ) throw ) [inline]`

An `auto_ptr` can be constructed from another `auto_ptr`.

**Parameters**

<code>__a</code>	Another <code>auto_ptr</code> of a different but related type.
------------------	--

A pointer-to-`Tp1` must be convertible to a pointer-to-`Tp/element_type`.

This object now *owns* the object previously owned by `__a`, which has given up ownership.

Definition at line 125 of file `auto_ptr.h`.

**4.499.3.4** `template<typename _Tp> std::auto_ptr<_Tp>::~~auto_ptr ( ) [inline]`

When the `auto_ptr` goes out of scope, the object it owns is deleted. If it no longer owns anything (i.e., `get ()` is `NULL`), then this has no effect.

The C++ standard says there is supposed to be an empty throw specification here, but omitting it is standard conforming. Its presence can be detected only if `_Tp::~~_Tp()` throws, but this is prohibited. [17.4.3.6]/2

Definition at line 170 of file `auto_ptr.h`.

**4.499.3.5** `template<typename _Tp> std::auto_ptr<_Tp>::auto_ptr ( auto_ptr_ref< element_type > __ref ) throw ) [inline]`

**Automatic conversions.**

These operations convert an `auto_ptr` into and from an `auto_ptr_ref` automatically as needed. This allows constructs such as

```
auto_ptr<Derived> func_returning_auto_ptr(...);
...
auto_ptr<Base> ptr = func_returning_auto_ptr(...);
```

Definition at line 260 of file `auto_ptr.h`.

**4.499.4 Member Function Documentation**

**4.499.4.1** `template<typename _Tp> element_type* std::auto_ptr<_Tp>::get ( ) const throw ) [inline]`

Bypassing the smart pointer.

**Returns**

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

## Note

This auto\_ptr still owns the memory.

Definition at line 211 of file auto\_ptr.h.

4.499.4.2 `template<typename _Tp> element_type& std::auto_ptr<_Tp>::operator*( ) const throw )` `[inline]`

Smart pointer dereferencing.

If this auto\_ptr no longer owns anything, then this operation will crash. (For a smart pointer, *no longer owns anything* is the same as being a null pointer, and you know what happens when you dereference one of those...)

Definition at line 181 of file auto\_ptr.h.

4.499.4.3 `template<typename _Tp> element_type* std::auto_ptr<_Tp>::operator->( ) const throw )` `[inline]`

Smart pointer dereferencing.

This returns the pointer itself, which the language then will automatically cause to be dereferenced.

Definition at line 194 of file auto\_ptr.h.

4.499.4.4 `template<typename _Tp> auto_ptr& std::auto_ptr<_Tp>::operator=( auto_ptr<_Tp> &__a ) throw )`  
`[inline]`

auto\_ptr assignment operator.

## Parameters

<code>__a</code>	Another auto_ptr of the same type.
------------------	------------------------------------

This object now *owns* the object previously owned by `__a`, which has given up ownership. The object that this one *used* to own and track has been deleted.

Definition at line 136 of file auto\_ptr.h.

References `std::auto_ptr<_Tp>::reset()`.

4.499.4.5 `template<typename _Tp> template<typename _Tp1> auto_ptr& std::auto_ptr<_Tp>::operator=( auto_ptr<_Tp1> &__a ) throw )` `[inline]`

auto\_ptr assignment operator.

## Parameters

<code>__a</code>	Another auto_ptr of a different but related type.
------------------	---

A pointer-to-Tp1 must be convertible to a pointer-to-Tp/element\_type.

This object now *owns* the object previously owned by `__a`, which has given up ownership. The object that this one *used* to own and track has been deleted.

Definition at line 154 of file auto\_ptr.h.

References `std::auto_ptr<_Tp>::reset()`.

4.499.4.6 `template<typename _Tp> element_type* std::auto_ptr<_Tp>::release( ) throw )` `[inline]`

Bypassing the smart pointer.

**Returns**

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

**Note**

This `auto_ptr` no longer owns the memory. When this object goes out of scope, nothing will happen.

Definition at line 225 of file `auto_ptr.h`.

4.499.4.7 `template<typename _Tp> void std::auto_ptr<_Tp>::reset ( element_type * __p = 0 ) throw ) [inline]`

Forcibly deletes the managed object.

**Parameters**

<code>__p</code>	A pointer (defaults to NULL).
------------------	-------------------------------

This object now *owns* the object pointed to by `__p`. The previous object has been deleted.

Definition at line 240 of file `auto_ptr.h`.

Referenced by `std::auto_ptr<_Tp>::operator=()`.

The documentation for this class was generated from the following file:

- [auto\\_ptr.h](#)

**4.500 std::auto\_ptr\_ref<\_Tp1> Struct Template Reference****Public Member Functions**

- `auto_ptr_ref (_Tp1 * __p)`

**Public Attributes**

- `_Tp1 * _M_ptr`

**4.500.1 Detailed Description**

`template<typename _Tp1> struct std::auto_ptr_ref<_Tp1>`

A wrapper class to provide `auto_ptr` with reference semantics. For example, an `auto_ptr` can be assigned (or constructed from) the result of a function which returns an `auto_ptr` by value.

All the `auto_ptr_ref` stuff should happen behind the scenes.

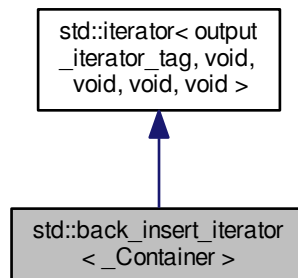
Definition at line 48 of file `auto_ptr.h`.

The documentation for this struct was generated from the following file:

- [auto\\_ptr.h](#)

## 4.501 std::back\_insert\_iterator&lt; \_Container &gt; Class Template Reference

Inheritance diagram for std::back\_insert\_iterator< \_Container >:



## Public Types

- typedef `_Container` `container_type`
- typedef void `difference_type`
- typedef `output_iterator_tag` `iterator_category`
- typedef void `pointer`
- typedef void `reference`
- typedef void `value_type`

## Public Member Functions

- `back_insert_iterator` (`_Container &__x`)
- `back_insert_iterator` & `operator*` ()
- `back_insert_iterator` & `operator++` ()
- `back_insert_iterator` `operator++` (int)
- `back_insert_iterator` & `operator=` (const typename `_Container::value_type` &\_\_value)
- `back_insert_iterator` & `operator=` (typename `_Container::value_type` &&\_\_value)

## Protected Attributes

- `_Container *` **`container`**

## 4.501.1 Detailed Description

```
template<typename _Container>class std::back_insert_iterator< _Container >
```

Turns assignment into insertion.

These are output iterators, constructed from a container-of-T. Assigning a T to the iterator appends it to the container using `push_back`.



Tip: Using the `back_inserter` function to create these iterators can save typing.

Definition at line 402 of file `stl_iterator.h`.

#### 4.501.2 Member Typedef Documentation

4.501.2.1 `template<typename _Container> typedef _Container std::back_insert_iterator<_Container>::container_type`

A nested typedef for the type of whatever container you used.

Definition at line 410 of file `stl_iterator.h`.

4.501.2.2 `typedef void std::iterator< output_iterator_tag, void, void, void, void>::difference_type` `[inherited]`

Distance between iterators is represented as this type.

Definition at line 125 of file `stl_iterator_base_types.h`.

4.501.2.3 `typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void>::iterator_category`  
`[inherited]`

One of the [tag types](#).

Definition at line 121 of file `stl_iterator_base_types.h`.

4.501.2.4 `typedef void std::iterator< output_iterator_tag, void, void, void, void>::pointer` `[inherited]`

This type represents a pointer-to-value\_type.

Definition at line 127 of file `stl_iterator_base_types.h`.

4.501.2.5 `typedef void std::iterator< output_iterator_tag, void, void, void, void>::reference` `[inherited]`

This type represents a reference-to-value\_type.

Definition at line 129 of file `stl_iterator_base_types.h`.

4.501.2.6 `typedef void std::iterator< output_iterator_tag, void, void, void, void>::value_type` `[inherited]`

The type "pointed to" by the iterator.

Definition at line 123 of file `stl_iterator_base_types.h`.

#### 4.501.3 Constructor & Destructor Documentation

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

The only way to create this iterator is with a container.

Definition at line 414 of file `stl_iterator.h`.

#### 4.501.4 Member Function Documentation

4.501.4.1 `template<typename _Container> back_insert_iterator& std::back_insert_iterator<_Container>::operator*( )` `[inline]`

Simply returns `*this`.

Definition at line 452 of file stl\_iterator.h.

```
4.501.4.2 template<typename _Container > back_insert_iterator& std::back_insert_iterator< _Container >::operator++ (
    ) [inline]
```

Simply returns \*this. (This iterator does not *move*.)

Definition at line 457 of file stl\_iterator.h.

```
4.501.4.3 template<typename _Container > back_insert_iterator std::back_insert_iterator< _Container >::operator++ (
    int ) [inline]
```

Simply returns \*this. (This iterator does not *move*.)

Definition at line 462 of file stl\_iterator.h.

```
4.501.4.4 template<typename _Container > back_insert_iterator& std::back_insert_iterator< _Container >::operator= (
    const typename _Container::value_type & __value ) [inline]
```

#### Parameters

<code>__value</code>	An instance of whatever type <code>container_type::const_reference</code> is; presumably a reference-to-const T for <code>container&lt;T&gt;</code> .
----------------------	---

#### Returns

This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the end, if you like). Assigning a value to the iterator will always append the value to the end of the container.

Definition at line 436 of file stl\_iterator.h.

The documentation for this class was generated from the following file:

- [stl\\_iterator.h](#)

## 4.502 std::bad\_weak\_ptr Class Reference

Inherits exception.

#### Public Member Functions

- virtual char const \* **what** () const noexcept

#### 4.502.1 Detailed Description

Exception possibly thrown by `shared_ptr`.

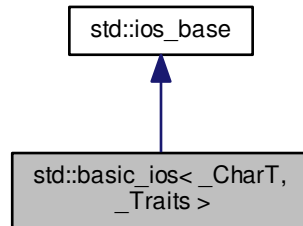
Definition at line 64 of file `shared_ptr_base.h`.

The documentation for this class was generated from the following file:

- [shared\\_ptr\\_base.h](#)

#### 4.503 `std::basic_ios< _CharT, _Traits >` Class Template Reference

Inheritance diagram for `std::basic_ios< _CharT, _Traits >`:



##### Public Types

- enum [event](#) { [erase\\_event](#), [imbue\\_event](#), [copyfmt\\_event](#) }
- typedef void(\* [event\\_callback](#)) ([event](#) \_\_e, [ios\\_base](#) &\_\_b, int \_\_i)
- typedef [\\_ios\\_Fmtflags](#) [fmtflags](#)
- typedef int [io\\_state](#)
- typedef [\\_ios\\_istate](#) [iostate](#)
- typedef int [open\\_mode](#)
- typedef [\\_ios\\_Openmode](#) [openmode](#)
- typedef int [seek\\_dir](#)
- typedef [\\_ios\\_Seekdir](#) [seekdir](#)
- typedef [std::streamoff](#) [streamoff](#)
- typedef [std::streampos](#) [streampos](#)
  
- typedef [\\_CharT](#) [char\\_type](#)
- typedef [\\_Traits::int\\_type](#) [int\\_type](#)
- typedef [\\_Traits::pos\\_type](#) [pos\\_type](#)
- typedef [\\_Traits::off\\_type](#) [off\\_type](#)
- typedef [\\_Traits](#) [traits\\_type](#)
  
- typedef [ctype< \\_CharT >](#) [\\_\\_ctype\\_type](#)
- typedef [num\\_put< \\_CharT, ostreambuf\\_iterator< \\_CharT, \\_Traits > >](#) [\\_\\_num\\_put\\_type](#)
- typedef [num\\_get< \\_CharT, istreambuf\\_iterator< \\_CharT, \\_Traits > >](#) [\\_\\_num\\_get\\_type](#)

##### Public Member Functions

- [basic\\_ios](#) ([basic\\_streambuf< \\_CharT, \\_Traits > \\*](#)\_\_sb)
- virtual [~basic\\_ios](#) ()
- const [locale](#) & [\\_M\\_getloc](#) () const
- void [\\_M\\_setstate](#) ([iostate](#) \_\_state)
- bool [bad](#) () const

- void `clear` (`iostate` \_\_state=`goodbit`)
- `basic_ios` & `copyfmt` (const `basic_ios` &\_\_rhs)
- bool `eof` () const
- `iostate` `exceptions` () const
- void `exceptions` (`iostate` \_\_except)
- bool `fail` () const
- `char_type` `fill` () const
- `char_type` `fill` (`char_type` \_\_ch)
- `fmtflags` `flags` () const
- `fmtflags` `flags` (`fmtflags` \_\_fmtfl)
- `locale` `getloc` () const
- bool `good` () const
- `locale` `imbue` (const `locale` &\_\_loc)
- long & `word` (int \_\_ix)
- char `narrow` (`char_type` \_\_c, char \_\_dfault) const
- `streamsize` `precision` () const
- `streamsize` `precision` (`streamsize` \_\_prec)
- void \*& `pword` (int \_\_ix)
- `basic_streambuf`< \_CharT, \_Traits > \* `rdbuf` () const
- `basic_streambuf`< \_CharT, \_Traits > \* `rdbuf` (`basic_streambuf`< \_CharT, \_Traits > \*\_\_sb)
- `iostate` `rdstate` () const
- void `register_callback` (`event_callback` \_\_fn, int \_\_index)
- `fmtflags` `setf` (`fmtflags` \_\_fmtfl)
- `fmtflags` `setf` (`fmtflags` \_\_fmtfl, `fmtflags` \_\_mask)
- void `setstate` (`iostate` \_\_state)
- `basic_ostream`< \_CharT, \_Traits > \* `tie` () const
- `basic_ostream`< \_CharT, \_Traits > \* `tie` (`basic_ostream`< \_CharT, \_Traits > \*\_\_tiestr)
- void `unsetf` (`fmtflags` \_\_mask)
- `char_type` `widen` (char \_\_c) const
- `streamsize` `width` () const
- `streamsize` `width` (`streamsize` \_\_wide)
- `operator void *` () const
- bool `operator!` () const

#### Static Public Member Functions

- static bool `sync_with_stdio` (bool \_\_sync=true)
- static int `xalloc` () throw ()

#### Static Public Attributes

- static const `fmtflags` `adjustfield`
- static const `openmode` `app`
- static const `openmode` `ate`
- static const `iostate` `badbit`
- static const `fmtflags` `basefield`
- static const `seekdir` `beg`
- static const `openmode` `binary`
- static const `fmtflags` `boolalpha`

- static const [seekdir](#) `cur`
- static const [fmtflags](#) `dec`
- static const [seekdir](#) `end`
- static const [iostate](#) `eofbit`
- static const [iostate](#) `failbit`
- static const [fmtflags](#) `fixed`
- static const [fmtflags](#) `floatfield`
- static const [iostate](#) `goodbit`
- static const [fmtflags](#) `hex`
- static const [openmode](#) `in`
- static const [fmtflags](#) `internal`
- static const [fmtflags](#) `left`
- static const [fmtflags](#) `oct`
- static const [openmode](#) `out`
- static const [fmtflags](#) `right`
- static const [fmtflags](#) `scientific`
- static const [fmtflags](#) `showbase`
- static const [fmtflags](#) `showpoint`
- static const [fmtflags](#) `showpos`
- static const [fmtflags](#) `skipws`
- static const [openmode](#) `trunc`
- static const [fmtflags](#) `unitbuf`
- static const [fmtflags](#) `uppercase`

#### Protected Types

- enum { `_S_local_word_size` }

#### Protected Member Functions

- [basic\\_ios](#) ()
- void `_M_cache_locale` (const [locale](#) &\_\_loc)
- void `_M_call_callbacks` ([event](#) \_\_ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- `_Words` & `_M_grow_words` (int \_\_index, bool \_\_iword)
- void `_M_init` () throw ()
- void `init` (basic\_streambuf< `_CharT`, `_Traits` > \*\_\_sb)

#### Protected Attributes

- `_Callback_list` \* `_M_callbacks`
- const [\\_\\_ctype\\_type](#) \* `_M_ctype`
- [iostate](#) `_M_exception`
- [char\\_type](#) `_M_fill`
- bool `_M_fill_init`
- [fmtflags](#) `_M_flags`
- [locale](#) `_M_ios_locale`
- `_Words` `_M_local_word` [`_S_local_word_size`]
- const [\\_\\_num\\_get\\_type](#) \* `_M_num_get`
- const [\\_\\_num\\_put\\_type](#) \* `_M_num_put`

- [streamsize \\_M\\_precision](#)
- [basic\\_streambuf< \\_CharT, \\_Traits > \\* \\_M\\_streambuf](#)
- [iostate \\_M\\_streambuf\\_state](#)
- [basic\\_ostream< \\_CharT, \\_Traits > \\* \\_M\\_tie](#)
- [streamsize \\_M\\_width](#)
- [\\_Words \\* \\_M\\_word](#)
- [int \\_M\\_word\\_size](#)
- [\\_Words \\_M\\_word\\_zero](#)

#### 4.503.1 Detailed Description

```
template<typename _CharT, typename _Traits>class std::basic_ios< _CharT, _Traits >
```

Template class basic\_ios, virtual base class for all stream classes.

##### Template Parameters

<code>_CharT</code>	Type of character stream.
<code>_Traits</code>	Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> .

Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar)`;) are consolidated in this class.

Definition at line 66 of file `basic_ios.h`.

#### 4.503.2 Member Typedef Documentation

4.503.2.1 `template<typename _CharT, typename _Traits > typedef ctype<_CharT> std::basic_ios< _CharT, _Traits >::__ctype_type`

These are non-standard types.

Definition at line 86 of file `basic_ios.h`.

4.503.2.2 `template<typename _CharT, typename _Traits > typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits >::__num_get_type`

These are non-standard types.

Definition at line 90 of file `basic_ios.h`.

4.503.2.3 `template<typename _CharT, typename _Traits > typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits >::__num_put_type`

These are non-standard types.

Definition at line 88 of file `basic_ios.h`.

4.503.2.4 `template<typename _CharT, typename _Traits > typedef _CharT std::basic_ios< _CharT, _Traits >::__char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 75 of file `basic_ios.h`.

4.503.2.5 `typedef void(* std::ios_base::event_callback)(event __e, ios_base &__b, int __i) [inherited]`

The type of an event callback function.

## Parameters

<code>__e</code>	One of the members of the event enum.
<code>__b</code>	Reference to the <code>ios_base</code> object.
<code>__i</code>	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 436 of file `ios_base.h`.

#### 4.503.2.6 `typedef _Ios_Fmtflags std::ios_base::fmtflags` `[inherited]`

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 255 of file `ios_base.h`.

#### 4.503.2.7 `template<typename _CharT, typename _Traits> typedef _Traits::int_type std::basic_ios<_CharT, _Traits>::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 76 of file `basic_ios.h`.

#### 4.503.2.8 typedef \_Ios\_Iostate std::ios\_base::iostate [inherited]

This is a bitmask type.

*\_Ios\_Iostate* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 330 of file ios\_base.h.

#### 4.503.2.9 template<typename \_CharT, typename \_Traits> typedef \_Traits::off\_type std::basic\_ios<\_CharT, \_Traits>::off\_type

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 78 of file basic\_ios.h.

#### 4.503.2.10 typedef \_Ios\_Openmode std::ios\_base::openmode [inherited]

This is a bitmask type.

*\_Ios\_Openmode* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

Definition at line 361 of file ios\_base.h.

#### 4.503.2.11 template<typename \_CharT, typename \_Traits> typedef \_Traits::pos\_type std::basic\_ios<\_CharT, \_Traits>::pos\_type

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 77 of file basic\_ios.h.

#### 4.503.2.12 typedef \_Ios\_Seekdir std::ios\_base::seekdir [inherited]

This is an enumerated type.

*\_Ios\_Seekdir* is implementation-defined. Defined values of type seekdir are:

- beg



- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

Definition at line 393 of file `ios_base.h`.

#### 4.503.2.13 `template<typename _CharT, typename _Traits> typedef _Traits std::basic_ios<_CharT, _Traits>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 79 of file `basic_ios.h`.

### 4.503.3 Member Enumeration Documentation

#### 4.503.3.1 `enum std::ios_base::event` [inherited]

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 419 of file `ios_base.h`.

### 4.503.4 Constructor & Destructor Documentation

#### 4.503.4.1 `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits>::basic_ios ( basic_streambuf<_CharT, _Traits> * __sb )` [inline], [explicit]

Constructor performs initialization.

The parameter is passed by derived streams.

Definition at line 264 of file `basic_ios.h`.

References `std::basic_ios<_CharT, _Traits>::init()`.

#### 4.503.4.2 `template<typename _CharT, typename _Traits> virtual std::basic_ios<_CharT, _Traits>::~basic_ios ( )` [inline], [virtual]

Empty.

The destructor does nothing. More specifically, it does not destroy the streambuf held by `rdbuf()`.

Definition at line 276 of file `basic_ios.h`.

#### 4.503.4.3 `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits>::basic_ios ( )` [inline], [protected]

Empty.

The default constructor does nothing and is not normally accessible to users.

Definition at line 454 of file `basic_ios.h`.

### 4.503.5 Member Function Documentation

4.503.5.1 `const locale& std::ios_base::_M_getloc ( ) const` `[inline],[inherited]`

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 706 of file ios\_base.h.

4.503.5.2 `template<typename _CharT, typename _Traits > bool std::basic_ios< _CharT, _Traits >::bad ( ) const`  
`[inline]`

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 205 of file basic\_ios.h.

References std::ios\_base::badbit, and std::basic\_ios< \_CharT, \_Traits >::rdstate().

4.503.5.3 `template<typename _CharT, typename _Traits > void std::basic_ios< _CharT, _Traits >::clear ( iostate __state = goodbit )`

[Re]sets the error state.

Parameters

<code>__state</code>	The new state flag(s) to set.
----------------------	-------------------------------

See std::ios\_base::iostate for the possible bit values. Most users will not need to pass an argument.

Referenced by std::basic\_ios< \_CharT, \_Traits >::exceptions(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

4.503.5.4 `template<typename _CharT, typename _Traits > basic_ios& std::basic_ios< _CharT, _Traits >::copyfmt ( const basic_ios< _CharT, _Traits > & __rhs )`

Copies fields of \_\_rhs into this.

Parameters

<code>__rhs</code>	The source values for the copies.
--------------------	-----------------------------------

Returns

Reference to this object.

All fields of \_\_rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase\_event. After copying, each (new) callback is invoked with copyfmt\_event. The final step is to copy exceptions().

4.503.5.5 `template<typename _CharT, typename _Traits > bool std::basic_ios< _CharT, _Traits >::eof ( ) const`  
`[inline]`

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 184 of file basic\_ios.h.

References `std::ios_base::eofbit`, and `std::basic_ios<_CharT, _Traits>::rdstate()`.

**4.503.5.6** `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits>::exceptions ( ) const`  
`[inline]`

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Definition at line 216 of file basic\_ios.h.

**4.503.5.7** `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits>::exceptions ( iostate`  
`__except ) [inline]`

Throwing exceptions on errors.

**Parameters**

<code>__except</code>	The new exceptions mask.
-----------------------	--------------------------

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 251 of file basic\_ios.h.

References `std::basic_ios<_CharT, _Traits>::clear()`.

**4.503.5.8** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::fail ( ) const`  
`[inline]`

Fast error checking.

**Returns**

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Definition at line 195 of file basic\_ios.h.

References std::ios\_base::badbit, std::ios\_base::failbit, and std::basic\_ios< \_CharT, \_Traits >::rdstate().

Referenced by std::basic\_ios< \_CharT, \_Traits >::operator void \*(), and std::basic\_ios< \_CharT, \_Traits >::operator!().

**4.503.5.9** `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill ( ) const`  
`[inline]`

Retrieves the *empty* character.

**Returns**

The current fill character.

It defaults to a space ( ' ' ) in the current locale.

Definition at line 364 of file basic\_ios.h.

References std::basic\_ios< \_CharT, \_Traits >::widen().

Referenced by std::basic\_ios< \_CharT, \_Traits >::fill().

**4.503.5.10** `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill ( char_type`  
`__ch ) [inline]`

Sets a new *empty* character.

**Parameters**

<code>__ch</code>	The new character.
-------------------	--------------------

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space ( ' ' ) in the current locale.

Definition at line 384 of file basic\_ios.h.

References std::basic\_ios< \_CharT, \_Traits >::fill().

**4.503.5.11** `fmtflags std::ios_base::flags ( ) const [inline],[inherited]`

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 551 of file ios\_base.h.

**4.503.5.12** `fmtflags std::ios_base::flags ( fmtflags __fmtfl ) [inline],[inherited]`

Setting new format flags all at once.

**Parameters**

<code>__fmtfl</code>	The new flags to set.
----------------------	-----------------------

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 562 of file `ios_base.h`.

**4.503.5.13** `locale std::ios_base::getloc ( ) const` `[inline]`, `[inherited]`

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 695 of file `ios_base.h`.

**4.503.5.14** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::good ( ) const` `[inline]`

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 174 of file `basic_ios.h`.

References `std::basic_ios<_CharT, _Traits>::rdstate()`.

**4.503.5.15** `template<typename _CharT, typename _Traits> locale std::basic_ios<_CharT, _Traits>::imbue ( const locale & __loc )`

Moves to a new locale.

**Parameters**

<code>__loc</code>	The new locale.
--------------------	-----------------

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

4.503.5.16 `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::init ( basic_streambuf< _CharT, _Traits > * __sb ) [protected]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Referenced by `std::basic_ios< _CharT, _Traits >::basic_ios()`.

4.503.5.17 `long& std::ios_base::iword ( int __ix ) [inline], [inherited]`

Access to integer array.

#### Parameters

<code>__ix</code>	Index into the array.
-------------------	-----------------------

#### Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 741 of file `ios_base.h`.

4.503.5.18 `template<typename _CharT, typename _Traits> char std::basic_ios< _CharT, _Traits >::narrow ( char_type __c, char __dfault ) const [inline]`

Squeezes characters.

#### Parameters

<code>__c</code>	The character to narrow.
<code>__dfault</code>	The character to narrow.

#### Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, dfault)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 424 of file `basic_ios.h`.

4.503.5.19 `template<typename _CharT, typename _Traits> std::basic_ios< _CharT, _Traits >::operator void * ( ) const [inline]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file `basic_ios.h`.

References `std::basic_ios<_CharT, _Traits>::fail()`.

**4.503.5.20** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::operator! ( ) const`  
`[inline]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 119 of file `basic_ios.h`.

References `std::basic_ios<_CharT, _Traits>::fail()`.

**4.503.5.21** `streamsize std::ios_base::precision ( ) const` `[inline], [inherited]`

Flags access.

#### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 621 of file `ios_base.h`.

**4.503.5.22** `streamsize std::ios_base::precision ( streamsize __prec )` `[inline], [inherited]`

Changing flags.

#### Parameters

<code>__prec</code>	The new precision value.
---------------------	--------------------------

#### Returns

The previous value of `precision()`.

Definition at line 630 of file `ios_base.h`.

**4.503.5.23** `void*& std::ios_base::pword ( int __ix )` `[inline], [inherited]`

Access to void pointer array.

#### Parameters

<code>__ix</code>	Index into the array.
-------------------	-----------------------

#### Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 762 of file `ios_base.h`.

4.503.5.24 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::rdbuf ( ) const [inline]`

Accessing the underlying buffer.

#### Returns

The current stream buffer.

This does not change the state of the stream.

Definition at line 315 of file `basic_ios.h`.

4.503.5.25 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::rdbuf ( basic_streambuf<_CharT, _Traits> * __sb )`

Changing the underlying buffer.

#### Parameters

<code>__sb</code>	The new stream buffer.
-------------------	------------------------

#### Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);           // ios == basic_ios<char>
```

4.503.5.26 `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits>::rdstate ( ) const [inline]`

Returns the error state of the stream buffer.

#### Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 131 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::bad()`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::basic_ios<_CharT, _Traits>::good()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

4.503.5.27 `void std::ios_base::register_callback ( event_callback __fn, int __index ) [inherited]`

Add the callback `__fn` with parameter `__index`.



## Parameters

<code>__fn</code>	The function to add.
<code>__index</code>	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**4.503.5.28** `fmtflags std::ios_base::setf ( fmtflags __fmtfl ) [inline],[inherited]`

Setting new format flags.

## Parameters

<code>__fmtfl</code>	Additional flags to set.
----------------------	--------------------------

## Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 578 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

**4.503.5.29** `fmtflags std::ios_base::setf ( fmtflags __fmtfl, fmtflags __mask ) [inline],[inherited]`

Setting new format flags.

## Parameters

<code>__fmtfl</code>	Additional flags to set.
<code>__mask</code>	The flags mask for <code>__fmtfl</code> .

## Returns

The previous format control flags.

This function clears `mask` in the format flags, then sets `__fmtfl & mask`. An example mask is `ios_base::adjustfield`.

Definition at line 595 of file `ios_base.h`.

**4.503.5.30** `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits>::setstate ( iostate __state ) [inline]`

Sets additional flags in the error state.

## Parameters

<code>__state</code>	The additional state flag(s) to set.
----------------------	--------------------------------------

See `std::ios_base::iostate` for the possible bit values.

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

References `std::basic_ios<_CharT, _Traits>::clear()`, and `std::basic_ios<_CharT, _Traits>::rdstate()`.

**4.503.5.31** `static bool std::ios_base::sync_with_stdio ( bool __sync = true ) [static],[inherited]`

Interaction with the standard C I/O objects.

## Parameters

<code>__sync</code>	Whether to synchronize or not.
---------------------	--------------------------------

## Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt1.html>

**4.503.5.32** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie ( ) const [inline]`

Fetches the current *tied* stream.

## Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 289 of file `basic_ios.h`.

**4.503.5.33** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie ( basic_ostream<_CharT, _Traits> * __tiestr ) [inline]`

Ties this stream to an output stream.

## Parameters

<code>__tiestr</code>	The output stream.
-----------------------	--------------------

## Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 301 of file `basic_ios.h`.

**4.503.5.34** `void std::ios_base::unsetf ( fmtflags __mask ) [inline], [inherited]`

Clearing format flags.

## Parameters

<code>__mask</code>	The flags to unset.
---------------------	---------------------

This function clears `__mask` in the format flags.

Definition at line 610 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

**4.503.5.35** `template<typename _CharT, typename _Traits> char_type std::basic_ios<_CharT, _Traits>::widen ( char __c ) const [inline]`

Widens characters.

**Parameters**

<code>__c</code>	The character to widen.
------------------	-------------------------

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 443 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::fill()`.

**4.503.5.36** `streamsize std::ios_base::width( ) const` `[inline], [inherited]`

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 644 of file `ios_base.h`.

**4.503.5.37** `streamsize std::ios_base::width( streamsize __wide )` `[inline], [inherited]`

Changing flags.

**Parameters**

<code>__wide</code>	The new width value.
---------------------	----------------------

**Returns**

The previous value of `width()`.

Definition at line 653 of file `ios_base.h`.

**4.503.5.38** `static int std::ios_base::xalloc( ) throw` `[static], [inherited]`

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

## 4.503.6 Member Data Documentation

## 4.503.6.1 const fmtflags std::ios\_base::adjustfield [static],[inherited]

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 310 of file `ios_base.h`.

Referenced by `std::internal()`, `std::left()`, and `std::right()`.

## 4.503.6.2 const openmode std::ios\_base::app [static],[inherited]

Seek to end before each write.

Definition at line 364 of file `ios_base.h`.

## 4.503.6.3 const openmode std::ios\_base::ate [static],[inherited]

Open and seek to end immediately after opening.

Definition at line 367 of file `ios_base.h`.

## 4.503.6.4 const iostate std::ios\_base::badbit [static],[inherited]

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 334 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::bad()`, and `std::basic_ios< _CharT, _Traits >::fail()`.

## 4.503.6.5 const fmtflags std::ios\_base::basefield [static],[inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 313 of file `ios_base.h`.

Referenced by `std::dec()`, `std::hex()`, and `std::oct()`.

## 4.503.6.6 const seekdir std::ios\_base::beg [static],[inherited]

Request a seek relative to the beginning of the stream.

Definition at line 396 of file `ios_base.h`.

## 4.503.6.7 const openmode std::ios\_base::binary [static],[inherited]

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 372 of file `ios_base.h`.

## 4.503.6.8 const fmtflags std::ios\_base::boolalpha [static],[inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 258 of file `ios_base.h`.

Referenced by `std::boolalpha()`, and `std::noboolalpha()`.

## 4.503.6.9 const seekdir std::ios\_base::cur [static],[inherited]

Request a seek relative to the current position within the sequence.

Definition at line 399 of file ios\_base.h.

**4.503.6.10** `const fmtflags std::ios_base::dec` `[static], [inherited]`

Converts integer input or generates integer output in decimal base.

Definition at line 261 of file ios\_base.h.

Referenced by `std::dec()`.

**4.503.6.11** `const seekdir std::ios_base::end` `[static], [inherited]`

Request a seek relative to the current end of the sequence.

Definition at line 402 of file ios\_base.h.

**4.503.6.12** `const iostate std::ios_base::eofbit` `[static], [inherited]`

Indicates that an input operation reached the end of an input sequence.

Definition at line 337 of file ios\_base.h.

Referenced by `std::basic_ios<_CharT, _Traits>::eof()`.

**4.503.6.13** `const iostate std::ios_base::failbit` `[static], [inherited]`

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 342 of file ios\_base.h.

Referenced by `std::basic_ios<_CharT, _Traits>::fail()`.

**4.503.6.14** `const fmtflags std::ios_base::fixed` `[static], [inherited]`

Generate floating-point output in fixed-point notation.

Definition at line 264 of file ios\_base.h.

Referenced by `std::fixed()`.

**4.503.6.15** `const fmtflags std::ios_base::floatfield` `[static], [inherited]`

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 316 of file ios\_base.h.

Referenced by `std::fixed()`, and `std::scientific()`.

**4.503.6.16** `const iostate std::ios_base::goodbit` `[static], [inherited]`

Indicates all is well.

Definition at line 345 of file ios\_base.h.

**4.503.6.17** `const fmtflags std::ios_base::hex` `[static], [inherited]`

Converts integer input or generates integer output in hexadecimal base.

Definition at line 267 of file ios\_base.h.

Referenced by `std::hex()`.

**4.503.6.18** `const openmode std::ios_base::in` `[static], [inherited]`

Open for input. Default for `ifstream` and `fstream`.

Definition at line 375 of file `ios_base.h`.

**4.503.6.19** `const fmtflags std::ios_base::internal` `[static], [inherited]`

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 272 of file `ios_base.h`.

Referenced by `std::internal()`.

**4.503.6.20** `const fmtflags std::ios_base::left` `[static], [inherited]`

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 276 of file `ios_base.h`.

Referenced by `std::left()`.

**4.503.6.21** `const fmtflags std::ios_base::oct` `[static], [inherited]`

Converts integer input or generates integer output in octal base.

Definition at line 279 of file `ios_base.h`.

Referenced by `std::oct()`.

**4.503.6.22** `const openmode std::ios_base::out` `[static], [inherited]`

Open for output. Default for `ofstream` and `fstream`.

Definition at line 378 of file `ios_base.h`.

**4.503.6.23** `const fmtflags std::ios_base::right` `[static], [inherited]`

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 283 of file `ios_base.h`.

Referenced by `std::right()`.

**4.503.6.24** `const fmtflags std::ios_base::scientific` `[static], [inherited]`

Generates floating-point output in scientific notation.

Definition at line 286 of file `ios_base.h`.

Referenced by `std::scientific()`.

**4.503.6.25** `const fmtflags std::ios_base::showbase` `[static], [inherited]`

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 290 of file `ios_base.h`.

Referenced by `std::noshowbase()`, and `std::showbase()`.

**4.503.6.26** `const fmtflags std::ios_base::showpoint` `[static], [inherited]`

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 294 of file `ios_base.h`.

Referenced by `std::noshowpoint()`, and `std::showpoint()`.

**4.503.6.27** `const fmtflags std::ios_base::showpos` `[static], [inherited]`

Generates a + sign in non-negative generated numeric output.

Definition at line 297 of file `ios_base.h`.

Referenced by `std::noshowpos()`, and `std::showpos()`.

**4.503.6.28** `const fmtflags std::ios_base::skipws` `[static], [inherited]`

Skips leading white space before certain input operations.

Definition at line 300 of file `ios_base.h`.

Referenced by `std::noskipws()`, and `std::skipws()`.

**4.503.6.29** `const openmode std::ios_base::trunc` `[static], [inherited]`

Open for input. Default for `ofstream`.

Definition at line 381 of file `ios_base.h`.

**4.503.6.30** `const fmtflags std::ios_base::unitbuf` `[static], [inherited]`

Flushes output after each output operation.

Definition at line 303 of file `ios_base.h`.

Referenced by `std::nounitbuf()`, and `std::unitbuf()`.

**4.503.6.31** `const fmtflags std::ios_base::uppercase` `[static], [inherited]`

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 307 of file `ios_base.h`.

Referenced by `std::nouppercase()`, and `std::uppercase()`.

The documentation for this class was generated from the following file:

- [basic\\_ios.h](#)

## **4.504** `std::basic_regex< _Ch_type, _Rx_traits >` Class Template Reference

### Public Types

- typedef [regex\\_constants::syntax\\_option\\_type](#) **flag\_type**
- typedef `traits_type::locale_type` **locale\_type**
- typedef `traits_type::string_type` **string\_type**
- typedef `_Rx_traits` **traits\_type**
- typedef `_Ch_type` **value\_type**

### Public Member Functions

- [basic\\_regex](#) ()

- `basic_regex` (const `_Ch_type` \* \_\_p, flag\_type \_\_f=ECMAScript)
- `basic_regex` (const `_Ch_type` \* \_\_p, std::size\_t \_\_len, flag\_type \_\_f)
- `basic_regex` (const `basic_regex` & \_\_rhs)
- `basic_regex` (const `basic_regex` && \_\_rhs) noexcept
- template<typename `_Ch_traits` , typename `_Ch_alloc` > `basic_regex` (const `std::basic_string`< `_Ch_type`, `_Ch_traits`, `_Ch_alloc` > & \_\_s, flag\_type \_\_f=ECMAScript)
- template<typename `_InputIterator` > `basic_regex` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, flag\_type \_\_f=ECMAScript)
- `basic_regex` (initializer\_list< `_Ch_type` > \_\_l, flag\_type \_\_f=ECMAScript)
- `~basic_regex` ()
- const `__detail::AutomatonPtr` & `_M_get_automaton` () const
- `basic_regex` & `assign` (const `basic_regex` & \_\_rhs)
- `basic_regex` & `assign` (`basic_regex` && \_\_rhs) noexcept
- `basic_regex` & `assign` (const `_Ch_type` \* \_\_p, flag\_type \_\_flags=ECMAScript)
- `basic_regex` & `assign` (const `_Ch_type` \* \_\_p, std::size\_t \_\_len, flag\_type \_\_flags)
- template<typename `_Ch_traits` , typename `_Alloc` > `basic_regex` & `assign` (const `basic_string`< `_Ch_type`, `_Ch_traits`, `_Alloc` > & \_\_s, flag\_type \_\_flags=ECMAScript)
- template<typename `_InputIterator` > `basic_regex` & `assign` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, flag\_type \_\_flags=ECMAScript)
- `basic_regex` & `assign` (initializer\_list< `_Ch_type` > \_\_l, flag\_type \_\_flags=ECMAScript)
- flag\_type `flags` () const
- locale\_type `getloc` () const
- locale\_type `imbue` (locale\_type \_\_loc)
- unsigned int `mark_count` () const
- `basic_regex` & `operator=` (const `basic_regex` & \_\_rhs)
- `basic_regex` & `operator=` (`basic_regex` && \_\_rhs) noexcept
- `basic_regex` & `operator=` (const `_Ch_type` \* \_\_p)
- template<typename `_Ch_traits` , typename `_Alloc` > `basic_regex` & `operator=` (const `basic_string`< `_Ch_type`, `_Ch_traits`, `_Alloc` > & \_\_s)
- void `swap` (`basic_regex` & \_\_rhs)

## Static Public Attributes

### Constants

*std* [28.8.1](1)

- static constexpr flag\_type `icase`
- static constexpr flag\_type `nosubs`
- static constexpr flag\_type `optimize`
- static constexpr flag\_type `collate`
- static constexpr flag\_type `ECMAScript`
- static constexpr flag\_type `basic`
- static constexpr flag\_type `extended`
- static constexpr flag\_type `awk`
- static constexpr flag\_type `grep`
- static constexpr flag\_type `egrep`

## Protected Attributes

- `__detail::AutomatonPtr` `_M_automaton`
- flag\_type `_M_flags`
- `_Rx_traits` `_M_traits`



## 4.504.1 Detailed Description

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> class std::basic_regex<_Ch_type, _Rx_traits>
```

Objects of specializations of this class represent regular expressions constructed from sequences of character type `_Ch_type`.

Storage for the regular expression is allocated and deallocated as necessary by the member functions of this class.

Definition at line 335 of file `regex.h`.

## 4.504.2 Constructor &amp; Destructor Documentation

```
4.504.2.1 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> std::basic_regex<_Ch_type,
    _Rx_traits>::basic_regex( ) [inline]
```

Constructs a basic regular expression that does not match any character sequence.

Definition at line 367 of file `regex.h`.

```
4.504.2.2 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> std::basic_regex<_Ch_type,
    _Rx_traits>::basic_regex( const _Ch_type * __p, flag_type __f = ECMAScript ) [inline], [explicit]
```

Constructs a basic regular expression from the sequence `[__p, __p + char_traits<_Ch_type>::length(__p))` interpreted according to the flags in `__f`.

## Parameters

<code>__p</code>	A pointer to the start of a C-style null-terminated string containing a regular expression.
<code>__f</code>	Flags indicating the syntax rules and options.

## Exceptions

<code>regex_error</code>	if <code>__p</code> is not a valid regular expression.
--------------------------	--

Definition at line 385 of file `regex.h`.

```
4.504.2.3 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> std::basic_regex<_Ch_type,
    _Rx_traits>::basic_regex( const _Ch_type * __p, std::size_t __len, flag_type __f ) [inline]
```

Constructs a basic regular expression from the sequence `[p, p + len)` interpreted according to the flags in `f`.

## Parameters

<code>__p</code>	A pointer to the start of a string containing a regular expression.
<code>__len</code>	The length of the string containing the regular expression.
<code>__f</code>	Flags indicating the syntax rules and options.

## Exceptions

<code>regex_error</code>	if <code>__p</code> is not a valid regular expression.
--------------------------	--

Definition at line 403 of file `regex.h`.

```
4.504.2.4 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> std::basic_regex<_Ch_type,
    _Rx_traits>::basic_regex( const basic_regex<_Ch_type, _Rx_traits> & __rhs ) [inline]
```

Copy-constructs a basic regular expression.

## Parameters

<code>__rhs</code>	A regex object.
--------------------	-----------------

Definition at line 413 of file regex.h.

```
4.504.2.5 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> std::basic_regex<_Ch_type,
_Rx_traits>::basic_regex ( const basic_regex<_Ch_type, _Rx_traits> && __rhs ) [inline],
[noexcept]
```

Move-constructs a basic regular expression.

## Parameters

<code>__rhs</code>	A regex object.
--------------------	-----------------

Definition at line 423 of file regex.h.

```
4.504.2.6 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> template<typename _Ch_traits ,
typename _Ch_alloc > std::basic_regex<_Ch_type, _Rx_traits>::basic_regex ( const std::basic_string<
_Ch_type, _Ch_traits, _Ch_alloc> & __s, flag_type __f = ECMAScript ) [inline], [explicit]
```

Constructs a basic regular expression from the string `s` interpreted according to the flags in `f`.

## Parameters

<code>__s</code>	A string containing a regular expression.
<code>__f</code>	Flags indicating the syntax rules and options.

## Exceptions

<code>regex_error</code>	if <code>__s</code> is not a valid regular expression.
--------------------------	--

Definition at line 439 of file regex.h.

```
4.504.2.7 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> template<typename _Inputiterator >
std::basic_regex<_Ch_type, _Rx_traits>::basic_regex ( _Inputiterator __first, _Inputiterator __last, flag_type __f
= ECMAScript ) [inline]
```

Constructs a basic regular expression from the range `[first, last)` interpreted according to the flags in `f`.

## Parameters

<code>__first</code>	The start of a range containing a valid regular expression.
<code>__last</code>	The end of a range containing a valid regular expression.
<code>__f</code>	The format flags of the regular expression.

## Exceptions

<code>regex_error</code>	if <code>[__first, __last)</code> is not a valid regular expression.
--------------------------	--

Definition at line 461 of file regex.h.

```
4.504.2.8 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> std::basic_regex<_Ch_type,
_Rx_traits>::basic_regex ( initializer_list<_Ch_type> __l, flag_type __f = ECMAScript ) [inline]
```

Constructs a basic regular expression from an initializer list.

## Parameters

<code>__l</code>	The initializer list.
<code>__f</code>	The format flags of the regular expression.

## Exceptions

<code>regex_error</code>	if <code>__l</code> is not a valid regular expression.
--------------------------	--

Definition at line 475 of file `regex.h`.

4.504.2.9 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> std::basic_regex<_Ch_type, _Rx_traits>::~basic_regex ( ) [inline]`

Destroys a basic regular expression.

Definition at line 485 of file `regex.h`.

## 4.504.3 Member Function Documentation

4.504.3.1 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex& std::basic_regex<_Ch_type, _Rx_traits>::assign ( const basic_regex<_Ch_type, _Rx_traits> &__rhs ) [inline]`

the real assignment operator.

## Parameters

<code>__rhs</code>	Another regular expression object.
--------------------	------------------------------------

Definition at line 531 of file `regex.h`.

References `std::basic_regex<_Ch_type, _Rx_traits>::swap()`.

Referenced by `std::basic_regex<_Ch_type, _Rx_traits>::assign()`, and `std::basic_regex<_Ch_type, _Rx_traits>::operator=()`.

4.504.3.2 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex& std::basic_regex<_Ch_type, _Rx_traits>::assign ( basic_regex<_Ch_type, _Rx_traits> &&__rhs ) [inline], [noexcept]`

The move-assignment operator.

## Parameters

<code>__rhs</code>	Another regular expression object.
--------------------	------------------------------------

Definition at line 544 of file `regex.h`.

References `std::move()`, and `std::basic_regex<_Ch_type, _Rx_traits>::swap()`.

4.504.3.3 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex& std::basic_regex<_Ch_type, _Rx_traits>::assign ( const _Ch_type * __p, flag_type __flags = ECMAScript ) [inline]`

Assigns a new regular expression to a regex object from a C-style null-terminated string containing a regular expression pattern.

## Parameters

---

<code>__p</code>	A pointer to a C-style null-terminated string containing a regular expression pattern.
<code>__flags</code>	Syntax option flags.

**Exceptions**

<code>regex_error</code>	if <code>__p</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex_error</code> is thrown, <code>*this</code> remains unchanged.
--------------------------	--

Definition at line 565 of file `regex.h`.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`.

**4.504.3.4** `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex& std::basic_regex<_Ch_type, _Rx_traits >::assign ( const _Ch_type * __p, std::size_t __len, flag_type __flags ) [inline]`

Assigns a new regular expression to a regex object from a C-style string containing a regular expression pattern.

**Parameters**

<code>__p</code>	A pointer to a C-style string containing a regular expression pattern.
<code>__len</code>	The length of the regular expression pattern string.
<code>__flags</code>	Syntax option flags.

**Exceptions**

<code>regex_error</code>	if <code>p</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex_error</code> is thrown, <code>*this</code> remains unchanged.
--------------------------	--

Definition at line 582 of file `regex.h`.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`.

**4.504.3.5** `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> template<typename _Ch_type_traits, typename _Alloc > basic_regex& std::basic_regex<_Ch_type, _Rx_traits >::assign ( const basic_string<_Ch_type, _Ch_type_traits, _Alloc > & __s, flag_type __flags = ECMAScript ) [inline]`

Assigns a new regular expression to a regex object from a string containing a regular expression pattern.

**Parameters**

<code>__s</code>	A string containing a regular expression pattern.
<code>__flags</code>	Syntax option flags.

**Exceptions**

<code>regex_error</code>	if <code>__s</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex_error</code> is thrown, <code>*this</code> remains unchanged.
--------------------------	--

Definition at line 598 of file `regex.h`.

References `std::basic_regex< _Ch_type, _Rx_traits >::swap()`.

**4.504.3.6** `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> template<typename _InputIterator > basic_regex& std::basic_regex<_Ch_type, _Rx_traits >::assign ( _InputIterator __first, _InputIterator __last, flag_type __flags = ECMAScript ) [inline]`

Assigns a new regular expression to a regex object.

**Parameters**

<code>__first</code>	The start of a range containing a valid regular expression.
<code>__last</code>	The end of a range containing a valid regular expression.
<code>__flags</code>	Syntax option flags.

**Exceptions**

<code>regex_error</code>	if <code>p</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex_error</code> is thrown, the object remains unchanged.
--------------------------	--

Definition at line 621 of file `regex.h`.

References `std::basic_regex<_Ch_type, _Rx_traits>::assign()`.

4.504.3.7 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex& std::basic_regex<_Ch_type, _Rx_traits>::assign ( initializer_list<_Ch_type> __l, flag_type __flags = ECMAScript ) [inline]`

Assigns a new regular expression to a regex object.

**Parameters**

<code>__l</code>	An initializer list representing a regular expression.
<code>__flags</code>	Syntax option flags.

**Exceptions**

<code>regex_error</code>	if <code>__l</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex_error</code> is thrown, the object remains unchanged.
--------------------------	--

Definition at line 637 of file `regex.h`.

References `std::basic_regex<_Ch_type, _Rx_traits>::assign()`.

4.504.3.8 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> flag_type std::basic_regex<_Ch_type, _Rx_traits>::flags ( ) const [inline]`

Gets the flags used to construct the regular expression or in the last call to `assign()`.

Definition at line 654 of file `regex.h`.

Referenced by `std::basic_regex<_Ch_type, _Rx_traits>::operator=()`.

4.504.3.9 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> locale_type std::basic_regex<_Ch_type, _Rx_traits>::getloc ( ) const [inline]`

Gets the locale currently imbued in the regular expression object.

Definition at line 672 of file `regex.h`.

4.504.3.10 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> locale_type std::basic_regex<_Ch_type, _Rx_traits>::imbue ( locale_type __loc ) [inline]`

Imbues the regular expression object with the given locale.

**Parameters**

<code>__loc</code>	A locale.
--------------------	-----------

Definition at line 664 of file regex.h.

4.504.3.11 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> unsigned int std::basic_regex<_Ch_type, _Rx_traits>::mark_count ( ) const [inline]`

Gets the number of marked subexpressions within the regular expression.

Definition at line 646 of file regex.h.

4.504.3.12 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex& std::basic_regex<_Ch_type, _Rx_traits>::operator= ( const basic_regex<_Ch_type, _Rx_traits> & __rhs ) [inline]`

Assigns one regular expression to another.

Definition at line 492 of file regex.h.

References `std::basic_regex<_Ch_type, _Rx_traits>::assign()`.

4.504.3.13 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex& std::basic_regex<_Ch_type, _Rx_traits>::operator= ( basic_regex<_Ch_type, _Rx_traits> && __rhs ) [inline], [noexcept]`

Move-assigns one regular expression to another.

Definition at line 499 of file regex.h.

References `std::basic_regex<_Ch_type, _Rx_traits>::assign()`, and `std::move()`.

4.504.3.14 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex& std::basic_regex<_Ch_type, _Rx_traits>::operator= ( const _Ch_type * __p ) [inline]`

Replaces a regular expression with a new one constructed from a C-style null-terminated string.

Parameters

<code>__p</code>	A pointer to the start of a null-terminated C-style string containing a regular expression.
------------------	---

Definition at line 510 of file regex.h.

References `std::basic_regex<_Ch_type, _Rx_traits>::assign()`, and `std::basic_regex<_Ch_type, _Rx_traits>::flags()`.

4.504.3.15 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> template<typename _Ch_type_traits, typename _Alloc> basic_regex& std::basic_regex<_Ch_type, _Rx_traits>::operator= ( const basic_string<_Ch_type, _Ch_type_traits, _Alloc> & __s ) [inline]`

Replaces a regular expression with a new one constructed from a string.

Parameters

<code>__s</code>	A pointer to a string containing a regular expression.
------------------	--

Definition at line 521 of file regex.h.

References `std::basic_regex<_Ch_type, _Rx_traits>::assign()`, and `std::basic_regex<_Ch_type, _Rx_traits>::flags()`.

4.504.3.16 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> void std::basic_regex<_Ch_type, _Rx_traits>::swap( basic_regex<_Ch_type, _Rx_traits> &__rhs ) [inline]`

Swaps the contents of two regular expression objects.

## Parameters

<code>__rhs</code>	Another regular expression object.
--------------------	------------------------------------

Definition at line 682 of file regex.h.

References `std::swap()`.

Referenced by `std::basic_regex< _Ch_type, _Rx_traits >::assign()`, and `std::swap()`.

The documentation for this class was generated from the following file:

- [regex.h](#)

## 4.505 std::basic\_string&lt; \_CharT, \_Traits, \_Alloc &gt; Class Template Reference

## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `__gnu_cxx::__normal_iterator< const_pointer, basic_string >` **const\_iterator**
- typedef `_CharT_alloc_type::const_pointer` **const\_pointer**
- typedef `_CharT_alloc_type::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_CharT_alloc_type::difference_type` **difference\_type**
- typedef `__gnu_cxx::__normal_iterator< pointer, basic_string >` **iterator**
- typedef `_CharT_alloc_type::pointer` **pointer**
- typedef `_CharT_alloc_type::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_CharT_alloc_type::size_type` **size\_type**
- typedef `_Traits` **traits\_type**
- typedef `_Traits::char_type` **value\_type**

## Public Member Functions

- `basic_string ()`
- `basic_string (const _Alloc &__a)`
- `basic_string (const basic_string &__str)`
- `basic_string (const basic_string &__str, size_type __pos, size_type __n=npos)`
- `basic_string (const basic_string &__str, size_type __pos, size_type __n, const _Alloc &__a)`
- `basic_string (const _CharT *__s, size_type __n, const _Alloc &__a=_Alloc())`
- `basic_string (const _CharT *__s, const _Alloc &__a=_Alloc())`
- `basic_string (size_type __n, _CharT __c, const _Alloc &__a=_Alloc())`
- `basic_string (basic_string &&__str) noexcept`
- `basic_string (initializer_list< _CharT > __l, const _Alloc &__a=_Alloc())`
- `template<class _InputIterator > basic_string (_InputIterator __beg, _InputIterator __end, const _Alloc &__a=_Alloc())`
- `~basic_string () noexcept`
- `basic_string & append (const basic_string &__str)`
- `basic_string & append (const basic_string &__str, size_type __pos, size_type __n)`
- `basic_string & append (const _CharT *__s, size_type __n)`
- `basic_string & append (const _CharT *__s)`
- `basic_string & append (size_type __n, _CharT __c)`
- `basic_string & append (initializer_list< _CharT > __l)`
- `template<class _InputIterator > basic_string & append (_InputIterator __first, _InputIterator __last)`



- [basic\\_string](#) & [assign](#) (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & [assign](#) ([basic\\_string](#) &&\_\_str)
- [basic\\_string](#) & [assign](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n)
- [basic\\_string](#) & [assign](#) (const \_CharT \*\_\_s, size\_type \_\_n)
- [basic\\_string](#) & [assign](#) (const \_CharT \*\_\_s)
- [basic\\_string](#) & [assign](#) (size\_type \_\_n, \_CharT \_\_c)
- template<class \_InputIterator > [basic\\_string](#) & [assign](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [basic\\_string](#) & [assign](#) (initializer\_list< \_CharT > \_\_l)
- const\_reference [at](#) (size\_type \_\_n) const
- reference [at](#) (size\_type \_\_n)
- reference [back](#) ()
- const\_reference [back](#) () const
- iterator [begin](#) () noexcept
- const\_iterator [begin](#) () const noexcept
- const \_CharT \* [c\\_str](#) () const noexcept
- size\_type [capacity](#) () const noexcept
- const\_iterator [cbegin](#) () const noexcept
- const\_iterator [cend](#) () const noexcept
- void [clear](#) () noexcept
- int [compare](#) (const [basic\\_string](#) &\_\_str) const
- int [compare](#) (size\_type \_\_pos, size\_type \_\_n, const [basic\\_string](#) &\_\_str) const
- int [compare](#) (size\_type \_\_pos1, size\_type \_\_n1, const [basic\\_string](#) &\_\_str, size\_type \_\_pos2, size\_type \_\_n2) const
- int [compare](#) (const \_CharT \*\_\_s) const
- int [compare](#) (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s) const
- int [compare](#) (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s, size\_type \_\_n2) const
- size\_type [copy](#) (\_CharT \*\_\_s, size\_type \_\_n, size\_type \_\_pos=0) const
- const\_reverse\_iterator [crbegin](#) () const noexcept
- const\_reverse\_iterator [crend](#) () const noexcept
- const \_CharT \* [data](#) () const noexcept
- bool [empty](#) () const noexcept
- iterator [end](#) () noexcept
- const\_iterator [end](#) () const noexcept
- [basic\\_string](#) & [erase](#) (size\_type \_\_pos=0, size\_type \_\_n=[npos](#))
- iterator [erase](#) (iterator \_\_position)
- iterator [erase](#) (iterator \_\_first, iterator \_\_last)
- size\_type [find](#) (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type [find](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos=0) const noexcept
- size\_type [find](#) (const \_CharT \*\_\_s, size\_type \_\_pos=0) const
- size\_type [find](#) (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- size\_type [find\\_first\\_not\\_of](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos=0) const noexcept
- size\_type [find\\_first\\_not\\_of](#) (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type [find\\_first\\_not\\_of](#) (const \_CharT \*\_\_s, size\_type \_\_pos=0) const
- size\_type [find\\_first\\_not\\_of](#) (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- size\_type [find\\_first\\_of](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos=0) const noexcept
- size\_type [find\\_first\\_of](#) (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type [find\\_first\\_of](#) (const \_CharT \*\_\_s, size\_type \_\_pos=0) const
- size\_type [find\\_first\\_of](#) (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- size\_type [find\\_last\\_not\\_of](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos=[npos](#)) const noexcept
- size\_type [find\\_last\\_not\\_of](#) (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type [find\\_last\\_not\\_of](#) (const \_CharT \*\_\_s, size\_type \_\_pos=[npos](#)) const

- size\_type [find\\_last\\_not\\_of](#) ( \_CharT \_\_c, size\_type \_\_pos=[npos](#)) const noexcept
- size\_type [find\\_last\\_of](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos=[npos](#)) const noexcept
- size\_type [find\\_last\\_of](#) (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type [find\\_last\\_of](#) (const \_CharT \* \_\_s, size\_type \_\_pos=[npos](#)) const
- size\_type [find\\_last\\_of](#) ( \_CharT \_\_c, size\_type \_\_pos=[npos](#)) const noexcept
- reference [front](#) ()
- const\_reference [front](#) () const
- allocator\_type [get\\_allocator](#) () const noexcept
- void [insert](#) (iterator \_\_p, size\_type \_\_n, \_CharT \_\_c)
- template<class \_InputIterator > void [insert](#) (iterator \_\_p, \_InputIterator \_\_beg, \_InputIterator \_\_end)
- void [insert](#) (iterator \_\_p, initializer\_list< \_CharT > \_\_l)
- [basic\\_string](#) & [insert](#) (size\_type \_\_pos1, const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & [insert](#) (size\_type \_\_pos1, const [basic\\_string](#) &\_\_str, size\_type \_\_pos2, size\_type \_\_n)
- [basic\\_string](#) & [insert](#) (size\_type \_\_pos, const \_CharT \* \_\_s, size\_type \_\_n)
- [basic\\_string](#) & [insert](#) (size\_type \_\_pos, const \_CharT \* \_\_s)
- [basic\\_string](#) & [insert](#) (size\_type \_\_pos, size\_type \_\_n, \_CharT \_\_c)
- iterator [insert](#) (iterator \_\_p, \_CharT \_\_c)
- size\_type [length](#) () const noexcept
- size\_type [max\\_size](#) () const noexcept
- [basic\\_string](#) & [operator+=](#) (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & [operator+=](#) (const \_CharT \* \_\_s)
- [basic\\_string](#) & [operator+=](#) ( \_CharT \_\_c)
- [basic\\_string](#) & [operator+=](#) (initializer\_list< \_CharT > \_\_l)
- [basic\\_string](#) & [operator=](#) (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & [operator=](#) (const \_CharT \* \_\_s)
- [basic\\_string](#) & [operator=](#) ( \_CharT \_\_c)
- [basic\\_string](#) & [operator=](#) ([basic\\_string](#) &&\_\_str)
- [basic\\_string](#) & [operator=](#) (initializer\_list< \_CharT > \_\_l)
- const\_reference [operator\[\]](#) (size\_type \_\_pos) const
- reference [operator\[\]](#) (size\_type \_\_pos)
- void [pop\\_back](#) ()
- void [push\\_back](#) ( \_CharT \_\_c)
- reverse\_iterator [rbegin](#) () noexcept
- const\_reverse\_iterator [rbegin](#) () const noexcept
- reverse\_iterator [rend](#) () noexcept
- const\_reverse\_iterator [rend](#) () const noexcept
- [basic\\_string](#) & [replace](#) (size\_type \_\_pos, size\_type \_\_n, const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & [replace](#) (size\_type \_\_pos1, size\_type \_\_n1, const [basic\\_string](#) &\_\_str, size\_type \_\_pos2, size\_type \_\_n2)
- [basic\\_string](#) & [replace](#) (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \* \_\_s, size\_type \_\_n2)
- [basic\\_string](#) & [replace](#) (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \* \_\_s)
- [basic\\_string](#) & [replace](#) (size\_type \_\_pos, size\_type \_\_n1, size\_type \_\_n2, \_CharT \_\_c)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const \_CharT \* \_\_s, size\_type \_\_n)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const \_CharT \* \_\_s)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, size\_type \_\_n, \_CharT \_\_c)
- template<class \_InputIterator > [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, \_InputIterator \_\_k1, \_InputIterator \_\_k2)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, \_CharT \* \_\_k1, \_CharT \* \_\_k2)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const \_CharT \* \_\_k1, const \_CharT \* \_\_k2)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, iterator \_\_k1, iterator \_\_k2)

- `basic_string` & `replace` (iterator \_\_i1, iterator \_\_i2, const\_iterator \_\_k1, const\_iterator \_\_k2)
- `basic_string` & `replace` (iterator \_\_i1, iterator \_\_i2, initializer\_list<\_CharT> \_\_l)
- void `reserve` (size\_type \_\_res\_arg=0)
- void `resize` (size\_type \_\_n, \_CharT \_\_c)
- void `resize` (size\_type \_\_n)
- size\_type `rfind` (const `basic_string` & \_\_str, size\_type \_\_pos=`npos`) const noexcept
- size\_type `rfind` (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type `rfind` (const \_CharT \* \_\_s, size\_type \_\_pos=`npos`) const
- size\_type `rfind` (\_CharT \_\_c, size\_type \_\_pos=`npos`) const noexcept
- void `shrink_to_fit` ()
- size\_type `size` () const noexcept
- `basic_string` `substr` (size\_type \_\_pos=0, size\_type \_\_n=`npos`) const
- void `swap` (`basic_string` & \_\_s)

#### Static Public Attributes

- static const size\_type `npos`

#### 4.505.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc> class std::basic_string<_CharT, _Traits, _Alloc>
```

Managing sequences of characters and character-like objects.

#### Template Parameters

<code>_CharT</code>	Type of character
<code>_Traits</code>	Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> .
<code>_Alloc</code>	Allocator type, defaults to <code>allocator&lt;_CharT&gt;</code> .

Meets the requirements of a `container`, a `reversible container`, and a `sequence`. Of the `optional sequence requirements`, only `push_back`, `at`, and `array access` are supported.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Documentation? What's that? Nathan Myers [ncm@cantrip.org](mailto:ncm@cantrip.org).

A string looks like this:

```

[basic_string<char_type>]
_M_dataplus
_M_p ----->
                                     [_Rep]
                                     _M_length
                                     _M_capacity
                                     _M_refcount
                                     unnamed array of char_type
```

Where the `_M_p` points to the first character in the string, and you cast it to a pointer-to-`_Rep` and subtract 1 to get a pointer to the header.

This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single `add` instruction: `_Rep::_M_data()`, and `string↔::_M_rep()`; and the allocation function which gets a block of raw bytes and with room enough and constructs a `_Rep` object at the front.

The reason you want `_M_data` pointing to the character array and not the `_Rep` is so that the debugger can see the string contents. (Probably we should add a non-inline member to get the `_Rep` for the debugger to use, so users can check the actual string length.)

Note that the `_Rep` object is a POD so that you can have a static *empty string* `_Rep` object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string `_Rep` object.

All but the last paragraph is considered pretty conventional for a C++ string implementation.

Definition at line 112 of file `basic_string.h`.

#### 4.505.2 Constructor & Destructor Documentation

4.505.2.1 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc >::basic_string( ) [inline]`

Default constructor creates an empty string.

Definition at line 437 of file `basic_string.h`.

Referenced by `std::basic_string< _CharT >::substr()`.

4.505.2.2 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc >::basic_string( const _Alloc & __a ) [explicit]`

Construct an empty string using allocator *a*.

4.505.2.3 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc >::basic_string( const basic_string< _CharT, _Traits, _Alloc > & __str )`

Construct string with copy of value of *str*.

Parameters

<code>__str</code>	Source string.
--------------------	----------------

4.505.2.4 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc >::basic_string( const basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos, size_type __n = npos )`

Construct string as copy of a substring.

Parameters

<code>__str</code>	Source string.
<code>__pos</code>	Index of first character to copy from.
<code>__n</code>	Number of characters to copy (default remainder).

4.505.2.5 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc >::basic_string( const basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos, size_type __n, const _Alloc & __a )`

Construct string as copy of a substring.

Parameters

<code>__str</code>	Source string.
<code>__pos</code>	Index of first character to copy from.

<code>__n</code>	Number of characters to copy.
<code>__a</code>	Allocator to use.

4.505.2.6 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc  
>::basic_string ( const _CharT * __s, size_type __n, const _Alloc & __a = _Alloc() )`

Construct string initialized by a character array.

Parameters

<code>__s</code>	Source character array.
<code>__n</code>	Number of characters to copy.
<code>__a</code>	Allocator to use (default is default allocator).

NB: `__s` must have at least `__n` characters, `'\0'` has no special meaning.

4.505.2.7 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc  
>::basic_string ( const _CharT * __s, const _Alloc & __a = _Alloc() )`

Construct string as copy of a C string.

Parameters

<code>__s</code>	Source C string.
<code>__a</code>	Allocator to use (default is default allocator).

4.505.2.8 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc  
>::basic_string ( size_type __n, _CharT __c, const _Alloc & __a = _Alloc() )`

Construct string as multiple characters.

Parameters

<code>__n</code>	Number of characters.
<code>__c</code>	Character to use.
<code>__a</code>	Allocator to use (default is default allocator).

4.505.2.9 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc  
>::basic_string ( basic_string< _CharT, _Traits, _Alloc > && __str ) [inline], [noexcept]`

Move construct string.

Parameters

<code>__str</code>	Source string.
--------------------	----------------

The newly-created string contains the exact contents of `__str`. `__str` is a valid, but unspecified string.

Definition at line 507 of file `basic_string.h`.

4.505.2.10 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc  
>::basic_string ( initializer_list< _CharT > __l, const _Alloc & __a = _Alloc() )`

Construct string from an initializer list.

## Parameters

<code>__l</code>	std::initializer_list of characters.
<code>__a</code>	Allocator to use (default is default allocator).

4.505.2.11 `template<typename _CharT, typename _Traits, typename _Alloc> template<class _InputIterator >  
std::basic_string< _CharT, _Traits, _Alloc >::basic_string ( _InputIterator __beg, _InputIterator __end, const  
_Alloc & __a = _Alloc() )`

Construct string as copy of a range.

## Parameters

<code>__beg</code>	Start of range.
<code>__end</code>	End of range.
<code>__a</code>	Allocator to use (default is default allocator).

4.505.2.12 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc  
>::~basic_string( ) [inline], [noexcept]`

Destroy the string instance.

Definition at line 538 of file basic\_string.h.

## 4.505.3 Member Function Documentation

4.505.3.1 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT,  
_Traits, _Alloc >::append ( const basic_string< _CharT, _Traits, _Alloc > & __str )`

Append a string to this string.

## Parameters

<code>__str</code>	The string to append.
--------------------	-----------------------

## Returns

Reference to this string.

Referenced by `std::basic_string< _CharT >::append()`, `std::operator+()`, and `std::basic_string< _CharT >::operator+=()`.

4.505.3.2 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT,  
_Traits, _Alloc >::append ( const basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos, size_type __n )`

Append a substring.

## Parameters

<code>__str</code>	The string to append.
<code>__pos</code>	Index of the first character of str to append.
<code>__n</code>	The number of characters to append.

## Returns

Reference to this string.

**Exceptions**

<i>std::out_of_range</i>	if <i>__pos</i> is not a valid index.
--------------------------	---------------------------------------

This function appends *\_\_n* characters from *\_\_str* starting at *\_\_pos* to this string. If *\_\_n* is larger than the number of available characters in *\_\_str*, the remainder of *\_\_str* is appended.

4.505.3.3 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::append ( const _CharT * __s, size_type __n )`

Append a C substring.

**Parameters**

<i>__s</i>	The C string to append.
<i>__n</i>	The number of characters to append.

**Returns**

Reference to this string.

4.505.3.4 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::append ( const _CharT * __s ) [inline]`

Append a C string.

**Parameters**

<i>__s</i>	The C string to append.
------------	-------------------------

**Returns**

Reference to this string.

Definition at line 1006 of file `basic_string.h`.

4.505.3.5 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::append ( size_type __n, _CharT __c )`

Append multiple characters.

**Parameters**

<i>__n</i>	The number of characters to append.
<i>__c</i>	The character to use.

**Returns**

Reference to this string.

Appends *\_\_n* copies of *\_\_c* to this string.

4.505.3.6 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::append ( initializer_list< _CharT > __l ) [inline]`

Append an `initializer_list` of characters.

## Parameters

<code>__l</code>	The initializer_list of characters to append.
------------------	---

## Returns

Reference to this string.

Definition at line 1030 of file basic\_string.h.

4.505.3.7 `template<typename _CharT, typename _Traits, typename _Alloc> template<class _InputIterator > basic_string& std::basic_string< _CharT, _Traits, _Alloc >::append ( _InputIterator __first, _InputIterator __last ) [inline]`

Append a range of characters.

## Parameters

<code>__first</code>	Iterator referencing the first character to append.
<code>__last</code>	Iterator marking the end of the range.

## Returns

Reference to this string.

Appends characters in the range [`__first`,`__last`) to this string.

Definition at line 1044 of file basic\_string.h.

4.505.3.8 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign ( const basic_string< _CharT, _Traits, _Alloc > & __str )`

Set value to contents of another string.

## Parameters

<code>__str</code>	Source string to use.
--------------------	-----------------------

## Returns

Reference to this string.

Referenced by `std::basic_string< _CharT >::assign()`, and `std::basic_string< _CharT >::operator=()`.

4.505.3.9 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign ( basic_string< _CharT, _Traits, _Alloc > && __str ) [inline]`

Set value to contents of another string.

## Parameters

<code>__str</code>	Source string to use.
--------------------	-----------------------

## Returns

Reference to this string.

This function sets this string to the exact contents of `__str`. `__str` is a valid, but unspecified string.

Definition at line 1079 of file basic\_string.h.



4.505.3.10 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::assign ( const basic_string<_CharT, _Traits, _Alloc> & __str, size_type __pos, size_type __n )`  
`[inline]`

Set value to a substring of a string.

## Parameters

<code>__str</code>	The string to use.
<code>__pos</code>	Index of the first character of str.
<code>__n</code>	Number of characters to use.

## Returns

Reference to this string.

## Exceptions

<code>std::out_of_range</code>	if <code>pos</code> is not a valid index.
--------------------------------	---

This function sets this string to the substring of `__str` consisting of `__n` characters at `__pos`. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is used.

Definition at line 1100 of file `basic_string.h`.

4.505.3.11 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign ( const _CharT * __s, size_type __n )`

Set value to a C substring.

## Parameters

<code>__s</code>	The C string to use.
<code>__n</code>	Number of characters to use.

## Returns

Reference to this string.

This function sets the value of this string to the first `__n` characters of `__s`. If `__n` is larger than the number of available characters in `__s`, the remainder of `__s` is used.

4.505.3.12 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign ( const _CharT * __s ) [inline]`

Set value to contents of a C string.

## Parameters

<code>__s</code>	The C string to use.
------------------	----------------------

## Returns

Reference to this string.

This function sets the value of this string to the value of `__s`. The data is copied, so there is no dependence on `__s` once the function returns.

Definition at line 1128 of file `basic_string.h`.

4.505.3.13 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign ( size_type __n, _CharT __c ) [inline]`

Set value to multiple characters.

**Parameters**

<code>__n</code>	Length of the resulting string.
<code>__c</code>	The character to use.

**Returns**

Reference to this string.

This function sets the value of this string to `__n` copies of character `__c`.

Definition at line 1144 of file `basic_string.h`.

4.505.3.14 `template<typename _CharT, typename _Traits, typename _Alloc> template<class _InputIterator > basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign ( _InputIterator __first, _InputIterator __last ) [inline]`

Set value to a range of characters.

**Parameters**

<code>__first</code>	Iterator referencing the first character to append.
<code>__last</code>	Iterator marking the end of the range.

**Returns**

Reference to this string.

Sets value of string to characters in the range `[__first,__last)`.

Definition at line 1157 of file `basic_string.h`.

4.505.3.15 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign ( initializer_list< _CharT > __l ) [inline]`

Set value to an `initializer_list` of characters.

**Parameters**

<code>__l</code>	The <code>initializer_list</code> of characters to assign.
------------------	--

**Returns**

Reference to this string.

Definition at line 1167 of file `basic_string.h`.

4.505.3.16 `template<typename _CharT, typename _Traits, typename _Alloc> const_reference std::basic_string< _CharT, _Traits, _Alloc >::at ( size_type __n ) const [inline]`

Provides access to the data contained in the string.

**Parameters**

<code>__n</code>	The index of the character to access.
------------------	---------------------------------------

**Returns**

Read-only (const) reference to the character.

**Exceptions**

<i>std::out_of_range</i>	If <i>n</i> is an invalid index.
--------------------------	----------------------------------

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

Definition at line 864 of file `basic_string.h`.

**4.505.3.17** `template<typename _CharT, typename _Traits, typename _Alloc> reference std::basic_string< _CharT, _Traits, _Alloc >::at( size_type __n ) [inline]`

Provides access to the data contained in the string.

**Parameters**

<code>__n</code>	The index of the character to access.
------------------	---------------------------------------

**Returns**

Read/write reference to the character.

**Exceptions**

<i>std::out_of_range</i>	If <i>n</i> is an invalid index.
--------------------------	----------------------------------

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails. Success results in unsharing the string.

Definition at line 883 of file `basic_string.h`.

**4.505.3.18** `template<typename _CharT, typename _Traits, typename _Alloc> reference std::basic_string< _CharT, _Traits, _Alloc >::back ( ) [inline]`

Returns a read/write reference to the data at the last element of the string.

Definition at line 913 of file `basic_string.h`.

**4.505.3.19** `template<typename _CharT, typename _Traits, typename _Alloc> const_reference std::basic_string< _CharT, _Traits, _Alloc >::back ( ) const [inline]`

Returns a read-only (constant) reference to the data at the last element of the string.

Definition at line 921 of file `basic_string.h`.

**4.505.3.20** `template<typename _CharT, typename _Traits, typename _Alloc> iterator std::basic_string< _CharT, _Traits, _Alloc >::begin ( ) [inline], [noexcept]`

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Definition at line 605 of file `basic_string.h`.

Referenced by `std::basic_string< _CharT >::crend()`, `std::regex_match()`, `std::regex_replace()`, `std::regex_search()`, and `std::basic_string< _CharT >::rend()`.

**4.505.3.21** `template<typename _CharT, typename _Traits, typename _Alloc> const_iterator std::basic_string< _CharT, _Traits, _Alloc >::begin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 616 of file `basic_string.h`.

**4.505.3.22** `template<typename _CharT, typename _Traits, typename _Alloc> const _CharT* std::basic_string< _CharT, _Traits, _Alloc >::c_str ( ) const [inline], [noexcept]`

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1800 of file basic\_string.h.

Referenced by std::operator==( ).

**4.505.3.23** `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::capacity ( ) const [inline], [noexcept]`

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 776 of file basic\_string.h.

Referenced by std::basic\_string< \_CharT >::push\_back(), and std::basic\_string< \_CharT >::shrink\_to\_fit().

**4.505.3.24** `template<typename _CharT, typename _Traits, typename _Alloc> const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 680 of file basic\_string.h.

**4.505.3.25** `template<typename _CharT, typename _Traits, typename _Alloc> const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cend ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 688 of file basic\_string.h.

**4.505.3.26** `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string< _CharT, _Traits, _Alloc >::clear ( ) [inline], [noexcept]`

Erases the string, making it empty.

Definition at line 803 of file basic\_string.h.

**4.505.3.27** `template<typename _CharT, typename _Traits, typename _Alloc> int std::basic_string< _CharT, _Traits, _Alloc >::compare ( const basic_string< _CharT, _Traits, _Alloc > & __str ) const [inline]`

Compare to a string.

#### Parameters

<code>__str</code>	String to compare against.
--------------------	----------------------------

#### Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before `__str`, 0 if their values are equivalent, or > 0 if this string is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 2225 of file basic\_string.h.

Referenced by std::sub\_match< \_Bi\_iter >::compare(), std::operator<(), std::operator<=(), std::operator==((), std::operator>(), and std::operator>=().

**4.505.3.28** template<typename \_CharT, typename \_Traits, typename \_Alloc> int std::basic\_string< \_CharT, \_Traits, \_Alloc >::compare ( size\_type \_\_pos, size\_type \_\_n, const basic\_string< \_CharT, \_Traits, \_Alloc > & \_\_str ) const

Compare substring to a string.

Parameters

<code>__pos</code>	Index of first character of substring.
<code>__n</code>	Number of characters in substring.
<code>__str</code>	String to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__str`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(),str.data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**4.505.3.29** template<typename \_CharT, typename \_Traits, typename \_Alloc> int std::basic\_string< \_CharT, \_Traits, \_Alloc >::compare ( size\_type \_\_pos1, size\_type \_\_n1, const basic\_string< \_CharT, \_Traits, \_Alloc > & \_\_str, size\_type \_\_pos2, size\_type \_\_n2 ) const

Compare substring to a substring.

Parameters

<code>__pos1</code>	Index of first character of substring.
<code>__n1</code>	Number of characters in substring.
<code>__str</code>	String to compare against.
<code>__pos2</code>	Index of first character of substring of str.
<code>__n2</code>	Number of characters in substring of str.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer < 0 if this substring is ordered before the substring of `__str`, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**4.505.3.30** template<typename \_CharT, typename \_Traits, typename \_Alloc> int std::basic\_string< \_CharT, \_Traits, \_Alloc >::compare ( const \_CharT \* \_\_s ) const

Compare to a C string.

**Parameters**

<code>__s</code>	C string to compare against.
------------------	------------------------------

**Returns**

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before `__s`, 0 if their values are equivalent, or > 0 if this string is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and the length of a string constructed from `__s`. The function then compares the two strings by calling `traits::compare(data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**4.505.3.31** `template<typename _CharT, typename _Traits, typename _Alloc> int std::basic_string<_CharT, _Traits, _Alloc>::compare ( size_type __pos, size_type __n1, const _CharT * __s ) const`

Compare substring to a C string.

**Parameters**

<code>__pos</code>	Index of first character of substring.
<code>__n1</code>	Number of characters in substring.
<code>__s</code>	C string to compare against.

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `pos`. Returns an integer < 0 if the substring is ordered before `__s`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and the length of a string constructed from `__s`. The function then compares the two string by calling `traits::compare(substring.data(),__s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**4.505.3.32** `template<typename _CharT, typename _Traits, typename _Alloc> int std::basic_string<_CharT, _Traits, _Alloc>::compare ( size_type __pos, size_type __n1, const _CharT * __s, size_type __n2 ) const`

Compare substring against a character array.

**Parameters**

<code>__pos</code>	Index of first character of substring.
<code>__n1</code>	Number of characters in substring.
<code>__s</code>	character array to compare against.
<code>__n2</code>	Number of characters of <code>s</code> .

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Form a string from the first `__n2` characters of `__s`. Returns an integer < 0 if this substring is ordered before the string from `__s`, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__n2`. The function then compares the two strings by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: `s` must have at least `n2` characters, `'\0'` has no special meaning.

4.505.3.33 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc>::copy ( _CharT * __s, size_type __n, size_type __pos = 0 ) const`

Copy substring into C string.



## Parameters

<code>__s</code>	C string to copy value into.
<code>__n</code>	Number of characters to copy.
<code>__pos</code>	Index of first character to copy.

## Returns

Number of characters actually copied

## Exceptions

<code>std::out_of_range</code>	If <code>__pos &gt; size()</code> .
--------------------------------	-------------------------------------

Copies up to `__n` characters starting at `__pos` into the C string `__s`. If `__pos` is greater than `size()`, `out_of_range` is thrown.

**4.505.3.34** `template<typename _CharT, typename _Traits, typename _Alloc> const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::crbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 697 of file `basic_string.h`.

**4.505.3.35** `template<typename _CharT, typename _Traits, typename _Alloc> const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::crend ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 706 of file `basic_string.h`.

**4.505.3.36** `template<typename _CharT, typename _Traits, typename _Alloc> const _CharT* std::basic_string<_CharT, _Traits, _Alloc>::data ( ) const [inline], [noexcept]`

Return const pointer to contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1810 of file `basic_string.h`.

Referenced by `std::basic_string<_CharT>::compare()`, `std::basic_string<_CharT>::find()`, `std::basic_string<_CharT>::find_first_not_of()`, `std::basic_string<_CharT>::find_last_of()`, `std::match_results<_FwdIterT, _Alloc>::format()`, and `std::regex_traits<_Ch_type>::transform()`.

**4.505.3.37** `template<typename _CharT, typename _Traits, typename _Alloc> bool std::basic_string<_CharT, _Traits, _Alloc>::empty ( ) const [inline], [noexcept]`

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 811 of file `basic_string.h`.

**4.505.3.38** `template<typename _CharT, typename _Traits, typename _Alloc> iterator std::basic_string<_CharT, _Traits, _Alloc>::end ( ) [inline], [noexcept]`

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Definition at line 624 of file `basic_string.h`.

Referenced by `std::basic_string<_CharT>::crbegin()`, `std::basic_string<_CharT>::rbegin()`, `std::regex_match()`, `std::regex_replace()`, and `std::regex_search()`.

4.505.3.39 `template<typename _CharT, typename _Traits, typename _Alloc> const_iterator std::basic_string<_CharT, _Traits, _Alloc>::end( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 635 of file basic\_string.h.

4.505.3.40 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::erase( size_type __pos = 0, size_type __n = npos ) [inline]`

Remove characters.

Parameters

<code>__pos</code>	Index of first character to remove (default 0).
<code>__n</code>	Number of characters to remove (default remainder).

Returns

Reference to this string.

Exceptions

<code>std::out_of_range</code>	If <code>pos</code> is beyond the end of this string.
--------------------------------	---

Removes `__n` characters from this string starting at `__pos`. The length of the string is reduced by `__n`. If there are `< __n` characters to remove, the remainder of the string is truncated. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1362 of file basic\_string.h.

Referenced by `std::basic_string<_CharT>::pop_back()`.

4.505.3.41 `template<typename _CharT, typename _Traits, typename _Alloc> iterator std::basic_string<_CharT, _Traits, _Alloc>::erase( iterator __position ) [inline]`

Remove one character.

Parameters

<code>__position</code>	Iterator referencing the character to remove.
-------------------------	---

Returns

iterator referencing same location after removal.

Removes the character at `__position` from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1378 of file basic\_string.h.

4.505.3.42 `template<typename _CharT, typename _Traits, typename _Alloc> iterator std::basic_string<_CharT, _Traits, _Alloc>::erase( iterator __first, iterator __last )`

Remove a range of characters.

Parameters

<code>__first</code>	Iterator referencing the first character to remove.
<code>__last</code>	Iterator referencing the end of the range.

**Returns**

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

4.505.3.43 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find ( const _CharT * __s, size_type __pos, size_type __n ) const`

Find position of a C substring.

**Parameters**

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to search from.
<code>__n</code>	Number of characters from <code>s</code> to search for.

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Referenced by `std::basic_string< _CharT >::find()`, and `std::basic_string< _CharT >::find_first_of()`.

4.505.3.44 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find ( const basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos = 0 ) const [inline], [noexcept]`

Find position of a string.

**Parameters**

<code>__str</code>	String to locate.
<code>__pos</code>	Index of character to search from (default 0).

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1846 of file `basic_string.h`.

4.505.3.45 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find ( const _CharT * __s, size_type __pos = 0 ) const [inline]`

Find position of a C string.

## Parameters

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to search from (default 0).

## Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1861 of file `basic_string.h`.

4.505.3.46 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find ( _CharT __c, size_type __pos = 0 ) const [noexcept]`

Find position of a character.

## Parameters

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search from (default 0).

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

4.505.3.47 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of ( const basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos = 0 ) const [inline], [noexcept]`

Find position of a character not in string.

## Parameters

<code>__str</code>	String containing characters to avoid.
<code>__pos</code>	Index of character to search from (default 0).

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2079 of file `basic_string.h`.

Referenced by `std::basic_string< _CharT >::find_first_not_of()`.

4.505.3.48 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of ( const _CharT * __s, size_type __pos, size_type __n ) const`

Find position of a character not in C substring.

**Parameters**

<code>__s</code>	C string containing characters to avoid.
<code>__pos</code>	Index of character to search from.
<code>__n</code>	Number of characters from <code>__s</code> to consider.

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

4.505.3.49 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of( const _CharT* __s, size_type __pos = 0 ) const [inline]`

Find position of a character not in C string.

**Parameters**

<code>__s</code>	C string containing characters to avoid.
<code>__pos</code>	Index of character to search from (default 0).

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2110 of file `basic_string.h`.

4.505.3.50 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of( _CharT __c, size_type __pos = 0 ) const [noexcept]`

Find position of a different character.

**Parameters**

<code>__c</code>	Character to avoid.
<code>__pos</code>	Index of character to search from (default 0).

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

4.505.3.51 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of( const basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos = 0 ) const [inline], [noexcept]`

Find position of a character of string.

## Parameters

<code>__str</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search from (default 0).

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1952 of file `basic_string.h`.

Referenced by `std::basic_string< _CharT >::find_first_of()`.

4.505.3.52 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of( const _CharT * __s, size_type __pos, size_type __n ) const`

Find position of a character of C substring.

## Parameters

<code>__s</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search from.
<code>__n</code>	Number of characters from <code>s</code> to search for.

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

4.505.3.53 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of( const _CharT * __s, size_type __pos = 0 ) const [inline]`

Find position of a character of C string.

## Parameters

<code>__s</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search from (default 0).

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1982 of file `basic_string.h`.

4.505.3.54 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of( _CharT __c, size_type __pos = 0 ) const [inline], [noexcept]`

Find position of a character.

## Parameters

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search from (default 0).

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for the character `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `find(__c, __pos)`.

Definition at line 2001 of file `basic_string.h`.

```
4.505.3.55  template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits,
            _Alloc >::find_last_not_of ( const basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos = npos ) const
            [inline], [noexcept]
```

Find last position of a character not in string.

## Parameters

<code>__str</code>	String containing characters to avoid.
<code>__pos</code>	Index of character to search back from (default end).

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2142 of file `basic_string.h`.

Referenced by `std::basic_string< _CharT >::find_last_not_of()`.

```
4.505.3.56  template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits,
            _Alloc >::find_last_not_of ( const _CharT * __s, size_type __pos, size_type __n ) const
```

Find last position of a character not in C substring.

## Parameters

<code>__s</code>	C string containing characters to avoid.
<code>__pos</code>	Index of character to search back from.
<code>__n</code>	Number of characters from <code>s</code> to consider.

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

```
4.505.3.57  template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits,
            _Alloc >::find_last_not_of ( const _CharT * __s, size_type __pos = npos ) const [inline]
```

Find last position of a character not in C string.

## Parameters

<code>__s</code>	C string containing characters to avoid.
<code>__pos</code>	Index of character to search back from (default end).

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2173 of file `basic_string.h`.

4.505.3.58 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of ( _CharT __c, size_type __pos = npos ) const [noexcept]`

Find last position of a different character.

## Parameters

<code>__c</code>	Character to avoid.
<code>__pos</code>	Index of character to search back from (default end).

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

4.505.3.59 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of ( const basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos = npos ) const [inline], [noexcept]`

Find last position of a character of string.

## Parameters

<code>__str</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search back from (default end).

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2016 of file `basic_string.h`.

Referenced by `std::basic_string< _CharT >::find_last_of()`.

4.505.3.60 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of ( const _CharT* __s, size_type __pos, size_type __n ) const`

Find last position of a character of C substring.



**Parameters**

<code>__s</code>	C string containing characters to locate.
<code>__pos</code>	Index of character to search back from.
<code>__n</code>	Number of characters from <code>s</code> to search for.

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

4.505.3.61 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of( const _CharT * __s, size_type __pos = npos ) const [inline]`

Find last position of a character of C string.

**Parameters**

<code>__s</code>	C string containing characters to locate.
<code>__pos</code>	Index of character to search back from (default end).

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2046 of file `basic_string.h`.

4.505.3.62 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of( _CharT __c, size_type __pos = npos ) const [inline], [noexcept]`

Find last position of a character.

**Parameters**

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search back from (default end).

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `rfind(__c, __pos)`.

Definition at line 2065 of file `basic_string.h`.

4.505.3.63 `template<typename _CharT, typename _Traits, typename _Alloc> reference std::basic_string< _CharT, _Traits, _Alloc >::front( ) [inline]`

Returns a read/write reference to the data at the first element of the string.

Definition at line 897 of file `basic_string.h`.

4.505.3.64 `template<typename _CharT, typename _Traits, typename _Alloc> const_reference std::basic_string<_CharT, _Traits, _Alloc>::front ( ) const [inline]`

Returns a read-only (constant) reference to the data at the first element of the string.

Definition at line 905 of file basic\_string.h.

4.505.3.65 `template<typename _CharT, typename _Traits, typename _Alloc> allocator_type std::basic_string<_CharT, _Traits, _Alloc>::get_allocator ( ) const [inline], [noexcept]`

Return copy of allocator used to construct this string.

Definition at line 1817 of file basic\_string.h.

Referenced by std::basic\_string<\_CharT>::basic\_string(), and std::basic\_string<\_CharT>::~~basic\_string().

4.505.3.66 `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string<_CharT, _Traits, _Alloc>::insert ( iterator __p, size_type __n, _CharT __c ) [inline]`

Insert multiple characters.

#### Parameters

<code>__p</code>	Iterator referencing location in string to insert at.
<code>__n</code>	Number of characters to insert
<code>__c</code>	The character to insert.

#### Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts `__n` copies of character `__c` starting at the position referenced by iterator `__p`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1185 of file basic\_string.h.

Referenced by std::basic\_string<\_CharT>::insert().

4.505.3.67 `template<typename _CharT, typename _Traits, typename _Alloc> template<class _InputIterator > void std::basic_string<_CharT, _Traits, _Alloc>::insert ( iterator __p, _InputIterator __beg, _InputIterator __end ) [inline]`

Insert a range of characters.

#### Parameters

<code>__p</code>	Iterator referencing location in string to insert at.
<code>__beg</code>	Start of range.
<code>__end</code>	End of range.

#### Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts characters in range `[__beg, __end)`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1202 of file basic\_string.h.

4.505.3.68 `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string<_CharT, _Traits, _Alloc>::insert ( iterator __p, initializer_list<_CharT> __l ) [inline]`

Insert an initializer\_list of characters.

## Parameters

<code>__p</code>	Iterator referencing location in string to insert at.
<code>__l</code>	The initializer_list of characters to insert.

## Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Definition at line 1213 of file `basic_string.h`.

4.505.3.69 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::insert( size_type __pos1, const basic_string<_CharT, _Traits, _Alloc> &__str ) [inline]`

Insert value of a string.

## Parameters

<code>__pos1</code>	Iterator referencing location in string to insert at.
<code>__str</code>	The string to insert.

## Returns

Reference to this string.

## Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts value of `__str` starting at `__pos1`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1233 of file `basic_string.h`.

4.505.3.70 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::insert( size_type __pos1, const basic_string<_CharT, _Traits, _Alloc> &__str, size_type __pos2, size_type __n ) [inline]`

Insert a substring.

## Parameters

<code>__pos1</code>	Iterator referencing location in string to insert at.
<code>__str</code>	The string to insert.
<code>__pos2</code>	Start of characters in <code>str</code> to insert.
<code>__n</code>	Number of characters to insert.

## Returns

Reference to this string.

## Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

<i>std::out_of_range</i>	If <i>pos1</i> > size() or <i>__pos2</i> > <i>str.size()</i> .
--------------------------	--

Starting at *pos1*, insert *\_\_n* character of *\_\_str* beginning with *\_\_pos2*. If adding characters causes the length to exceed *max\_size()*, *length\_error* is thrown. If *\_\_pos1* is beyond the end of this string or *\_\_pos2* is beyond the end of *\_\_str*, *out\_of\_range* is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1255 of file *basic\_string.h*.

4.505.3.71 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::insert ( size_type __pos, const _CharT * __s, size_type __n )`

Insert a C substring.

#### Parameters

<i>__pos</i>	Iterator referencing location in string to insert at.
<i>__s</i>	The C string to insert.
<i>__n</i>	The number of characters to insert.

#### Returns

Reference to this string.

#### Exceptions

<i>std::length_error</i>	If new length exceeds <i>max_size()</i> .
<i>std::out_of_range</i>	If <i>__pos</i> is beyond the end of this string.

Inserts the first *\_\_n* characters of *\_\_s* starting at *\_\_pos*. If adding characters causes the length to exceed *max\_size()*, *length\_error* is thrown. If *\_\_pos* is beyond *end()*, *out\_of\_range* is thrown. The value of the string doesn't change if an error is thrown.

4.505.3.72 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::insert ( size_type __pos, const _CharT * __s ) [inline]`

Insert a C string.

#### Parameters

<i>__pos</i>	Iterator referencing location in string to insert at.
<i>__s</i>	The C string to insert.

#### Returns

Reference to this string.

#### Exceptions

<i>std::length_error</i>	If new length exceeds <i>max_size()</i> .
<i>std::out_of_range</i>	If <i>pos</i> is beyond the end of this string.

Inserts the first *n* characters of *\_\_s* starting at *\_\_pos*. If adding characters causes the length to exceed *max\_size()*, *length\_error* is thrown. If *\_\_pos* is beyond *end()*, *out\_of\_range* is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1296 of file *basic\_string.h*.

4.505.3.73 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::insert ( size_type __pos, size_type __n, _CharT __c ) [inline]`

Insert multiple characters.

## Parameters

<code>__pos</code>	Index in string to insert at.
<code>__n</code>	Number of characters to insert
<code>__c</code>	The character to insert.

## Returns

Reference to this string.

## Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
<code>std::out_of_range</code>	If <code>__pos</code> is beyond the end of this string.

Inserts `__n` copies of character `__c` starting at index `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos > length()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1319 of file `basic_string.h`.

4.505.3.74 `template<typename _CharT, typename _Traits, typename _Alloc> iterator std::basic_string<_CharT, _Traits, _Alloc>::insert( iterator __p, _CharT __c ) [inline]`

Insert one character.

## Parameters

<code>__p</code>	Iterator referencing position in string to insert at.
<code>__c</code>	The character to insert.

## Returns

Iterator referencing newly inserted char.

## Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts character `__c` at position referenced by `__p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1337 of file `basic_string.h`.

4.505.3.75 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc>::length( ) const [inline], [noexcept]`

Returns the number of characters in the string, not including any null-termination.

Definition at line 721 of file `basic_string.h`.

4.505.3.76 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc>::max_size( ) const [inline], [noexcept]`

Returns the `size()` of the largest possible string.

Definition at line 726 of file `basic_string.h`.

4.505.3.77 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::operator+=( const basic_string<_CharT, _Traits, _Alloc> &__str ) [inline]`

Append a string to this string.

## Parameters

<code>__str</code>	The string to append.
--------------------	-----------------------

## Returns

Reference to this string.

Definition at line 932 of file basic\_string.h.

4.505.3.78 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator+=( const _CharT * __s ) [inline]`

Append a C string.

## Parameters

<code>__s</code>	The C string to append.
------------------	-------------------------

## Returns

Reference to this string.

Definition at line 941 of file basic\_string.h.

4.505.3.79 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator+=( _CharT __c ) [inline]`

Append a character.

## Parameters

<code>__c</code>	The character to append.
------------------	--------------------------

## Returns

Reference to this string.

Definition at line 950 of file basic\_string.h.

4.505.3.80 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator+=( initializer_list< _CharT > __l ) [inline]`

Append an initializer\_list of characters.

## Parameters

<code>__l</code>	The initializer_list of characters to be appended.
------------------	--

## Returns

Reference to this string.

Definition at line 963 of file basic\_string.h.

4.505.3.81 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator=( const basic_string< _CharT, _Traits, _Alloc > & __str ) [inline]`

Assign the value of *str* to this string.



## Parameters

<code>__str</code>	Source string.
--------------------	----------------

Definition at line 546 of file `basic_string.h`.

4.505.3.82 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator= ( const _CharT * __s ) [inline]`

Copy contents of `s` into this string.

## Parameters

<code>__s</code>	Source null-terminated string.
------------------	--------------------------------

Definition at line 554 of file `basic_string.h`.

4.505.3.83 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator= ( _CharT __c ) [inline]`

Set value to string of length 1.

## Parameters

<code>__c</code>	Source character.
------------------	-------------------

Assigning to a character makes this string length 1 and `(*this)[0] == c`.

Definition at line 565 of file `basic_string.h`.

4.505.3.84 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator= ( basic_string< _CharT, _Traits, _Alloc > && __str ) [inline]`

Move assign the value of `str` to this string.

## Parameters

<code>__str</code>	Source string.
--------------------	----------------

The contents of `str` are moved into this string (without copying). `str` is a valid, but unspecified string.

Definition at line 580 of file `basic_string.h`.

4.505.3.85 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator= ( initializer_list< _CharT > __l ) [inline]`

Set value to string constructed from initializer list.

## Parameters

<code>__l</code>	<code>std::initializer_list</code> .
------------------	--------------------------------------

Definition at line 592 of file `basic_string.h`.

4.505.3.86 `template<typename _CharT, typename _Traits, typename _Alloc> const_reference std::basic_string< _CharT, _Traits, _Alloc >::operator[] ( size_type __pos ) const [inline]`

Subscript access to the data contained in the string.

## Parameters

<code>__pos</code>	The index of the character to access.
--------------------	---------------------------------------

## Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 826 of file `basic_string.h`.

Referenced by `std::basic_string< _CharT >::back()`, and `std::basic_string< _CharT >::front()`.

**4.505.3.87** `template<typename _CharT, typename _Traits, typename _Alloc> reference std::basic_string< _CharT, _Traits, _Alloc >::operator[]( size_type __pos ) [inline]`

Subscript access to the data contained in the string.

## Parameters

<code>__pos</code>	The index of the character to access.
--------------------	---------------------------------------

## Returns

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.) Unshares the string.

Definition at line 843 of file `basic_string.h`.

**4.505.3.88** `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string< _CharT, _Traits, _Alloc >::pop_back( ) [inline]`

Remove the last character.

The string must be non-empty.

Definition at line 1407 of file `basic_string.h`.

**4.505.3.89** `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string< _CharT, _Traits, _Alloc >::push_back( _CharT __c ) [inline]`

Append a single character.

## Parameters

<code>__c</code>	Character to append.
------------------	----------------------

Definition at line 1052 of file `basic_string.h`.

Referenced by `std::basic_string< _CharT >::operator+=( )`.

**4.505.3.90** `template<typename _CharT, typename _Traits, typename _Alloc> reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rbegin( ) [inline], [noexcept]`

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 644 of file `basic_string.h`.

**4.505.3.91** `template<typename _CharT, typename _Traits, typename _Alloc> const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::rbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 653 of file `basic_string.h`.

**4.505.3.92** `template<typename _CharT, typename _Traits, typename _Alloc> reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::rend ( ) [inline], [noexcept]`

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 662 of file `basic_string.h`.

**4.505.3.93** `template<typename _CharT, typename _Traits, typename _Alloc> const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::rend ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 671 of file `basic_string.h`.

**4.505.3.94** `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace ( size_type __pos, size_type __n, const basic_string<_CharT, _Traits, _Alloc> & __str ) [inline]`

Replace characters with value from another string.

#### Parameters

<code>__pos</code>	Index of first character to replace.
<code>__n</code>	Number of characters to be replaced.
<code>__str</code>	String to insert.

#### Returns

Reference to this string.

#### Exceptions

<code>std::out_of_range</code>	If <code>pos</code> is beyond the end of this string.
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[__pos, __pos+__n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1429 of file `basic_string.h`.

Referenced by `std::basic_string<_CharT>::append()`, `std::basic_string<_CharT>::assign()`, `std::basic_string<_CharT>::insert()`, and `std::basic_string<_CharT>::replace()`.

**4.505.3.95** `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace ( size_type __pos1, size_type __n1, const basic_string<_CharT, _Traits, _Alloc> & __str, size_type __pos2, size_type __n2 ) [inline]`

Replace characters with value from another string.

## Parameters

<code>__pos1</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__str</code>	String to insert.
<code>__pos2</code>	Index of first character of str to use.
<code>__n2</code>	Number of characters from str to use.

## Returns

Reference to this string.

## Exceptions

<code>std::out_of_range</code>	If <code>__pos1 &gt; size()</code> or <code>__pos2 &gt; __str.size()</code> .
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[__pos1, __pos1 + n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1451 of file `basic_string.h`.

4.505.3.96 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace ( size_type __pos, size_type __n1, const _CharT * __s, size_type __n2 )`

Replace characters with value of a C substring.

## Parameters

<code>__pos</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__s</code>	C string to insert.
<code>__n2</code>	Number of characters from s to use.

## Returns

Reference to this string.

## Exceptions

<code>std::out_of_range</code>	If <code>pos1 &gt; size()</code> .
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[__pos, __pos + __n1)` from this string. In place, the first `__n2` characters of `__s` are inserted, or all of `__s` if `__n2` is too large. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

4.505.3.97 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace ( size_type __pos, size_type __n1, const _CharT * __s ) [inline]`

Replace characters with value of a C string.

## Parameters

<code>__pos</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__s</code>	C string to insert.

**Returns**

Reference to this string.

**Exceptions**

<code>std::out_of_range</code>	If <code>pos &gt; size()</code> .
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[__pos, __pos + __n1)` from this string. In place, the characters of `__s` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1496 of file `basic_string.h`.

```
4.505.3.98 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT,
    _Traits, _Alloc >::replace ( size_type __pos, size_type __n1, size_type __n2, _CharT __c ) [inline]
```

Replace characters with multiple characters.

**Parameters**

<code>__pos</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__n2</code>	Number of characters to insert.
<code>__c</code>	Character to insert.

**Returns**

Reference to this string.

**Exceptions**

<code>std::out_of_range</code>	If <code>__pos &gt; size()</code> .
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[pos, pos + n1)` from this string. In place, `__n2` copies of `__c` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1520 of file `basic_string.h`.

```
4.505.3.99 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT,
    _Traits, _Alloc >::replace ( iterator __i1, iterator __i2, const basic_string< _CharT, _Traits, _Alloc > & __str )
    [inline]
```

Replace range of characters with string.

**Parameters**

<code>__i1</code>	Iterator referencing start of range to replace.
-------------------	---

<code>__i2</code>	Iterator referencing end of range to replace.
<code>__str</code>	String value to insert.

**Returns**

Reference to this string.

**Exceptions**

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[__i1, __i2)`. In place, the value of `__str` is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1538 of file `basic_string.h`.

4.505.3.100 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace ( iterator __i1, iterator __i2, const _CharT* __s, size_type __n ) [inline]`

Replace range of characters with C substring.

**Parameters**

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__s</code>	C string value to insert.
<code>__n</code>	Number of characters from <code>s</code> to insert.

**Returns**

Reference to this string.

**Exceptions**

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[__i1, __i2)`. In place, the first `__n` characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1557 of file `basic_string.h`.

4.505.3.101 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace ( iterator __i1, iterator __i2, const _CharT* __s ) [inline]`

Replace range of characters with C string.

**Parameters**

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__s</code>	C string value to insert.

**Returns**

Reference to this string.

**Exceptions**

<i>std::length_error</i>	If new length exceeds <code>max_size()</code> .
--------------------------	---

Removes the characters in the range `[__i1,__i2)`. In place, the characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1578 of file `basic_string.h`.

4.505.3.102 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace ( iterator __i1, iterator __i2, size_type __n, _CharT __c ) [inline]`

Replace range of characters with multiple characters.

**Parameters**

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__n</code>	Number of characters to insert.
<code>__c</code>	Character to insert.

**Returns**

Reference to this string.

**Exceptions**

<i>std::length_error</i>	If new length exceeds <code>max_size()</code> .
--------------------------	---

Removes the characters in the range `[__i1,__i2)`. In place, `__n` copies of `__c` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1599 of file `basic_string.h`.

4.505.3.103 `template<typename _CharT, typename _Traits, typename _Alloc> template<class _InputIterator > basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace ( iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2 ) [inline]`

Replace range of characters with range.

**Parameters**

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__k1</code>	Iterator referencing start of range to insert.
<code>__k2</code>	Iterator referencing end of range to insert.

**Returns**

Reference to this string.

**Exceptions**

<i>std::length_error</i>	If new length exceeds <code>max_size()</code> .
--------------------------	---

Removes the characters in the range `[__i1,__i2)`. In place, characters in the range `[__k1,__k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1623 of file `basic_string.h`.

4.505.3.104 template<typename \_CharT, typename \_Traits, typename \_Alloc> basic\_string& std::basic\_string<\_CharT, \_Traits, \_Alloc>::replace ( iterator \_\_i1, iterator \_\_i2, initializer\_list<\_CharT> \_\_l ) [inline]

Replace range of characters with initializer\_list.



**Parameters**

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__l</code>	The initializer_list of characters to insert.

**Returns**

Reference to this string.

**Exceptions**

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[__i1,__i2)`. In place, characters in the range `[__k1,__k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1692 of file `basic_string.h`.

4.505.3.105 `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string< _CharT, _Traits, _Alloc >::reserve ( size_type __res_arg = 0 )`

Attempt to preallocate enough memory for specified number of characters.

**Parameters**

<code>__res_arg</code>	Number of characters required.
------------------------	--------------------------------

**Exceptions**

<code>std::length_error</code>	If <code>__res_arg</code> exceeds <code>max_size()</code> .
--------------------------------	---

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

Referenced by `std::basic_string< _CharT >::push_back()`, and `std::basic_string< _CharT >::shrink_to_fit()`.

4.505.3.106 `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string< _CharT, _Traits, _Alloc >::resize ( size_type __n, _CharT __c )`

Resizes the string to the specified number of characters.

**Parameters**

<code>__n</code>	Number of characters the string should contain.
<code>__c</code>	Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to `__c`.

Referenced by `std::basic_string< _CharT >::resize()`.

4.505.3.107 `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string< _CharT, _Traits, _Alloc >::resize ( size_type __n ) [inline]`

Resizes the string to the specified number of characters.

## Parameters

<code>__n</code>	Number of characters the string should contain.
------------------	---

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as char, this means setting them to 0.

Definition at line 753 of file basic\_string.h.

4.505.3.108 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc>::rfind ( const basic_string<_CharT, _Traits, _Alloc> & __str, size_type __pos = npos ) const [inline], [noexcept]`

Find last position of a string.

## Parameters

<code>__str</code>	String to locate.
<code>__pos</code>	Index of character to search back from (default end).

## Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1891 of file basic\_string.h.

Referenced by `std::basic_string<_CharT>::find_last_of()`, and `std::basic_string<_CharT>::rfind()`.

4.505.3.109 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc>::rfind ( const _CharT * __s, size_type __pos, size_type __n ) const`

Find last position of a C substring.

## Parameters

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to search back from.
<code>__n</code>	Number of characters from s to search for.

## Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

4.505.3.110 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc>::rfind ( const _CharT * __s, size_type __pos = npos ) const [inline]`

Find last position of a C string.

## Parameters

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to start search at (default end).

## Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1921 of file `basic_string.h`.

4.505.3.111 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc>::rfind ( _CharT __c, size_type __pos = npos ) const [noexcept]`

Find last position of a character.

## Parameters

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search back from (default end).

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

4.505.3.112 `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string<_CharT, _Traits, _Alloc>::shrink_to_fit ( ) [inline]`

A non-binding request to `reduce_capacity()` to `size()`.

Definition at line 759 of file `basic_string.h`.

4.505.3.113 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc>::size ( ) const [inline], [noexcept]`

Returns the number of characters in the string, not including any null-termination.

Definition at line 715 of file `basic_string.h`.

Referenced by `std::basic_string<_CharT>::assign()`, `std::basic_string<_CharT>::at()`, `std::basic_string<_CharT>::back()`, `std::basic_string<_CharT>::cend()`, `std::basic_string<_CharT>::clear()`, `std::basic_string<_CharT>::compare()`, `std::basic_string<_CharT>::empty()`, `std::basic_string<_CharT>::end()`, `std::basic_string<_CharT>::find()`, `std::basic_string<_CharT>::find_first_not_of()`, `std::basic_string<_CharT>::find_last_of()`, `std::match_results<_FwdIterT, _Alloc>::format()`, `std::basic_string<_CharT>::insert()`, `std::basic_string<_CharT>::operator[]()`, `std::basic_string<_CharT>::pop_back()`, `std::basic_string<_CharT>::push_back()`, `std::basic_string<_CharT>::replace()`, `std::basic_string<_CharT>::shrink_to_fit()`, and `std::regex_traits<_Ch_type>::transform()`.

4.505.3.114 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string std::basic_string<_CharT, _Traits, _Alloc>::substr ( size_type __pos = 0, size_type __n = npos ) const [inline]`

Get a substring.

## Parameters

<code>__pos</code>	Index of first character (default 0).
<code>__n</code>	Number of characters in substring (default remainder).

## Returns

The new string.

## Exceptions

<code>std::out_of_range</code>	If <code>__pos &gt; size()</code> .
--------------------------------	-------------------------------------

Construct and return a new string using the `__n` characters starting at `__pos`. If the string is too short, use the remainder of the characters. If `__pos` is beyond the end of the string, `out_of_range` is thrown.

Definition at line 2206 of file `basic_string.h`.

4.505.3.115 `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string< _CharT, _Traits, _Alloc >::swap ( basic_string< _CharT, _Traits, _Alloc > & __s )`

Swap contents with another string.

## Parameters

<code>__s</code>	String to swap with.
------------------	----------------------

Exchanges the contents of this string with that of `__s` in constant time.

Referenced by `std::basic_string< _CharT >::assign()`, `std::basic_string< _CharT >::operator=()`, and `std::swap()`.

## 4.505.4 Member Data Documentation

4.505.4.1 `template<typename _CharT, typename _Traits, typename _Alloc> const size_type std::basic_string< _CharT, _Traits, _Alloc >::npos [static]`

Value returned by various member functions when they fail.

Definition at line 285 of file `basic_string.h`.

The documentation for this class was generated from the following file:

- [basic\\_string.h](#)

## 4.506 std::bernoulli\_distribution Class Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef bool [result\\_type](#)

## Public Member Functions

- [bernoulli\\_distribution](#) (double `__p`=0.5)

- **bernoulli\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename [\\_ForwardIterator](#) , typename [\\_UniformRandomNumberGenerator](#) > void **\_\_generate** ([\\_ForwardIterator](#) \_\_f, [\\_ForwardIterator](#) \_\_t, [\\_UniformRandomNumberGenerator](#) &\_\_urng)
- template<typename [\\_ForwardIterator](#) , typename [\\_UniformRandomNumberGenerator](#) > void **\_\_generate** ([\\_ForwardIterator](#) \_\_f, [\\_ForwardIterator](#) \_\_t, [\\_UniformRandomNumberGenerator](#) &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename [\\_UniformRandomNumberGenerator](#) > void **\_\_generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, [\\_UniformRandomNumberGenerator](#) &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) max () const
- [result\\_type](#) min () const
- template<typename [\\_UniformRandomNumberGenerator](#) > [result\\_type](#) operator() ([\\_UniformRandomNumberGenerator](#) &\_\_urng)
- template<typename [\\_UniformRandomNumberGenerator](#) > [result\\_type](#) operator() ([\\_UniformRandomNumberGenerator](#) &\_\_urng, const [param\\_type](#) &\_\_p)
- double [p](#) () const
- [param\\_type](#) [param](#) () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()

#### Friends

- bool operator== (const [bernoulli\\_distribution](#) &\_\_d1, const [bernoulli\\_distribution](#) &\_\_d2)

#### 4.506.1 Detailed Description

A Bernoulli random number distribution.

Generates a sequence of true and false values with likelihood  $p$  that true will come up and  $(1 - p)$  that false will appear.

Definition at line 3572 of file random.h.

#### 4.506.2 Member Typedef Documentation

##### 4.506.2.1 typedef bool std::bernoulli\_distribution::result\_type

The type of the range of the distribution.

Definition at line 3576 of file random.h.

#### 4.506.3 Constructor & Destructor Documentation

##### 4.506.3.1 std::bernoulli\_distribution::bernoulli\_distribution ( double \_\_p = 0.5 ) [inline], [explicit]

Constructs a Bernoulli distribution with likelihood  $p$ .

#### Parameters

<a href="#">__p</a>	[IN] The likelihood of a true result being returned. Must be in the interval $[0, 1]$ .
---------------------	---

Definition at line 3609 of file random.h.

## 4.506.4 Member Function Documentation

4.506.4.1 **result\_type** std::bernoulli\_distribution::max ( ) const [inline]

Returns the least upper bound value of the distribution.

Definition at line 3659 of file random.h.

References std::max().

4.506.4.2 **result\_type** std::bernoulli\_distribution::min ( ) const [inline]

Returns the greatest lower bound value of the distribution.

Definition at line 3652 of file random.h.

References std::min().

4.506.4.3 **template**<typename \_UniformRandomNumberGenerator > **result\_type** std::bernoulli\_distribution::operator() ( \_UniformRandomNumberGenerator & \_\_urng ) [inline]

Generating functions.

Definition at line 3667 of file random.h.

4.506.4.4 **double** std::bernoulli\_distribution::p ( ) const [inline]

Returns the *p* parameter of the distribution.

Definition at line 3630 of file random.h.

4.506.4.5 **param\_type** std::bernoulli\_distribution::param ( ) const [inline]

Returns the parameter set of the distribution.

Definition at line 3637 of file random.h.

Referenced by std::operator>>().

4.506.4.6 **void** std::bernoulli\_distribution::param ( const param\_type & \_\_param ) [inline]

Sets the parameter set of the distribution.

Parameters

<b>__param</b>	The new parameter set of the distribution.
----------------	--

Definition at line 3645 of file random.h.

4.506.4.7 **void** std::bernoulli\_distribution::reset ( ) [inline]

Resets the distribution state.

Does nothing for a Bernoulli distribution.

Definition at line 3624 of file random.h.

## 4.506.5 Friends And Related Function Documentation

4.506.5.1 **bool** operator== ( const bernoulli\_distribution & \_\_d1, const bernoulli\_distribution & \_\_d2 ) [friend]

Return true if two Bernoulli distributions have the same parameters.

Definition at line 3709 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

#### 4.507 std::bernoulli\_distribution::param\_type Struct Reference

##### Public Types

- typedef [bernoulli\\_distribution](#) **distribution\_type**

##### Public Member Functions

- **param\_type** (double \_\_p=0.5)
- double **p** () const

##### Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

##### 4.507.1 Detailed Description

Parameter type.

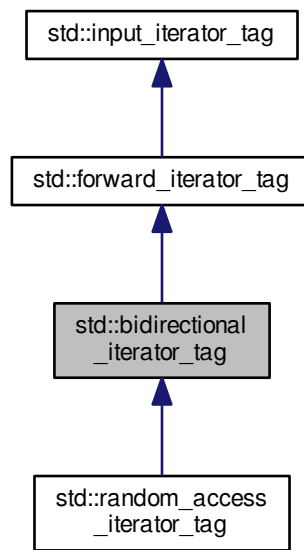
Definition at line 3578 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 4.508 `std::bidirectional_iterator_tag` Struct Reference

Inheritance diagram for `std::bidirectional_iterator_tag`:



### 4.508.1 Detailed Description

Bidirectional iterators support a superset of forward iterator operations.

Definition at line 99 of file `stl_iterator_base_types.h`.

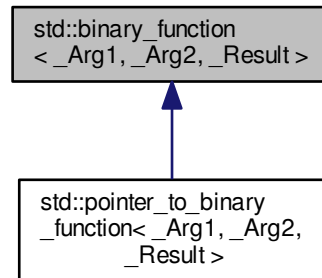
The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)



#### 4.509 std::binary\_function< \_Arg1, \_Arg2, \_Result > Struct Template Reference

Inheritance diagram for std::binary\_function< \_Arg1, \_Arg2, \_Result >:



##### Public Types

- typedef \_Arg1 [first\\_argument\\_type](#)
- typedef \_Result [result\\_type](#)
- typedef \_Arg2 [second\\_argument\\_type](#)

##### 4.509.1 Detailed Description

```
template<typename _Arg1, typename _Arg2, typename _Result> struct std::binary_function< _Arg1, _Arg2, _Result >
```

This is one of the [functor base classes](#).

Definition at line 114 of file stl\_function.h.

##### 4.509.2 Member Typedef Documentation

4.509.2.1 `template<typename _Arg1, typename _Arg2, typename _Result> typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result >::first_argument_type`

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl\_function.h.

4.509.2.2 `template<typename _Arg1, typename _Arg2, typename _Result> typedef _Result std::binary_function< _Arg1, _Arg2, _Result >::result_type`

`result_type` is the return type

Definition at line 123 of file stl\_function.h.

4.509.2.3 `template<typename _Arg1, typename _Arg2, typename _Result> typedef _Arg2 std::binary_function<_Arg1, _Arg2, _Result>::second_argument_type`

`second_argument_type` is the type of the second argument

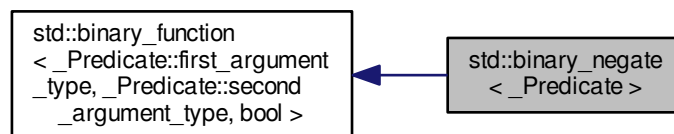
Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.510 `std::binary_negate<_Predicate>` Class Template Reference

Inheritance diagram for `std::binary_negate<_Predicate>`:



### Public Types

- `typedef _Predicate::first_argument_type` [first\\_argument\\_type](#)
- `typedef bool` [result\\_type](#)
- `typedef _Predicate::second_argument_type` [second\\_argument\\_type](#)

### Public Member Functions

- **`binary_negate`** (`const _Predicate &__x`)
- **`operator()`** (`const typename _Predicate::first_argument_type &__x, const typename _Predicate::second_argument_type &__y`) `const`

### Protected Attributes

- `_Predicate` **`M_pred`**

### 4.510.1 Detailed Description

`template<typename _Predicate> class std::binary_negate<_Predicate>`

One of the [negation functors](#).

Definition at line 374 of file `stl_function.h`.

#### 4.510.2 Member Typedef Documentation

4.510.2.1 `typedef _Predicate::first_argument_type std::binary_function< _Predicate::first_argument_type ,  
_Predicate::second_argument_type , bool >::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.510.2.2 `typedef bool std::binary_function< _Predicate::first_argument_type , _Predicate::second_argument_type , bool  
>::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.510.2.3 `typedef _Predicate::second_argument_type std::binary_function< _Predicate::first_argument_type ,  
_Predicate::second_argument_type , bool >::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

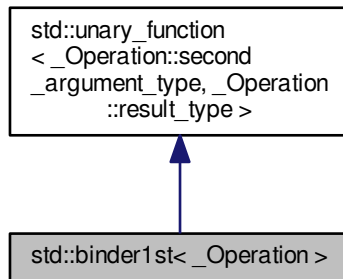
Definition at line 120 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

#### 4.511 `std::binder1st< _Operation >` Class Template Reference

Inheritance diagram for `std::binder1st< _Operation >`:



#### Public Types

- `typedef _Operation::second_argument_type` [argument\\_type](#)
- `typedef _Operation::result_type` [result\\_type](#)

## Public Member Functions

- **binder1st** (const `_Operation` &\_\_x, const typename `_Operation::first_argument_type` &\_\_y)
- `_Operation::result_type` **operator()** (const typename `_Operation::second_argument_type` &\_\_x) const
- `_Operation::result_type` **operator()** (typename `_Operation::second_argument_type` &\_\_x) const

## Protected Attributes

- `_Operation` **op**
- `_Operation::first_argument_type` **value**

### 4.511.1 Detailed Description

`template<typename _Operation>class std::binder1st<_Operation>`

One of the [binder functors](#).

Definition at line 104 of file `binders.h`.

### 4.511.2 Member Typedef Documentation

4.511.2.1 `typedef _Operation::second_argument_type std::unary_function<_Operation::second_argument_type ,  
_Operation::result_type>::argument_type` `[inherited]`

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

4.511.2.2 `typedef _Operation::result_type std::unary_function<_Operation::second_argument_type ,_Operation::result_type  
>::result_type` `[inherited]`

`result_type` is the return type

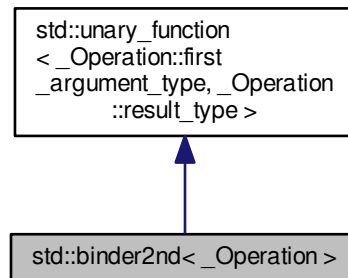
Definition at line 107 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [binders.h](#)

#### 4.512 std::binder2nd< \_Operation > Class Template Reference

Inheritance diagram for std::binder2nd< \_Operation >:



##### Public Types

- typedef `_Operation::first_argument_type` [argument\\_type](#)
- typedef `_Operation::result_type` [result\\_type](#)

##### Public Member Functions

- **binder2nd** (const `_Operation` &\_\_x, const typename `_Operation::second_argument_type` &\_\_y)
- `_Operation::result_type` **operator()** (const typename `_Operation::first_argument_type` &\_\_x) const
- `_Operation::result_type` **operator()** (typename `_Operation::first_argument_type` &\_\_x) const

##### Protected Attributes

- `_Operation` **op**
- `_Operation::second_argument_type` **value**

##### 4.512.1 Detailed Description

template<typename `_Operation`>class std::binder2nd< `_Operation` >

One of the [binder functors](#).

Definition at line 139 of file `binders.h`.

##### 4.512.2 Member Typedef Documentation

4.512.2.1 typedef `_Operation::first_argument_type` `std::unary_function<_Operation::first_argument_type, _Operation::result_type>::argument_type` [\[inherited\]](#)

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

4.512.2.2 `typedef _Operation::result_type std::unary_function<_Operation::first_argument_type, _Operation::result_type>::result_type` [inherited]

`result_type` is the return type

Definition at line 107 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [binders.h](#)

## 4.513 `std::binomial_distribution<_IntType>` Class Template Reference

### Classes

- struct [param\\_type](#)

### Public Types

- `typedef _IntType` [result\\_type](#)

### Public Member Functions

- **`binomial_distribution`** (`_IntType __t=_IntType(1), double __p=0.5`)
- **`binomial_distribution`** (`const` [param\\_type](#) &`__p`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator> void __generate` (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator> void __generate` (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const` [param\\_type](#) &`__p`)
- `template<typename _UniformRandomNumberGenerator> void __generate` (`result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const` [param\\_type](#) &`__p`)
- [result\\_type](#) `max` () `const`
- [result\\_type](#) `min` () `const`
- `template<typename _UniformRandomNumberGenerator> result_type operator()` (`_UniformRandomNumberGenerator &__urng`)
- `template<typename _UniformRandomNumberGenerator> result_type operator()` (`_UniformRandomNumberGenerator &__urng, const` [param\\_type](#) &`__p`)
- `double` `p` () `const`
- [param\\_type](#) `param` () `const`
- `void` `param` (`const` [param\\_type](#) &`__param`)
- `void` `reset` ()
- `_IntType` `t` () `const`

### Friends

- `template<typename _IntType1, typename _CharT, typename _Traits> std::basic_ostream<_CharT, _Traits> &operator<<` (`std::basic_ostream<_CharT, _Traits> &__os, const` `std::binomial_distribution<_IntType1>` &`__x`)
- `bool` `operator==` (`const` `binomial_distribution` &`__d1, const` `binomial_distribution` &`__d2`)
- `template<typename _IntType1, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits> &operator>>` (`std::basic_istream<_CharT, _Traits> &__is, std::binomial_distribution<_IntType1>` &`__x`)

## 4.513.1 Detailed Description

```
template<typename _IntType = int> class std::binomial_distribution< _IntType >
```

A discrete binomial random number distribution.

The formula for the binomial probability density function is  $p(i|t, p) = \binom{t}{i} p^i (1-p)^{t-i}$  where  $t$  and  $p$  are the parameters of the distribution.

Definition at line 3777 of file random.h.

## 4.513.2 Member Typedef Documentation

```
4.513.2.1 template<typename _IntType = int> typedef _IntType std::binomial_distribution< _IntType >::result_type
```

The type of the range of the distribution.

Definition at line 3780 of file random.h.

## 4.513.3 Member Function Documentation

```
4.513.3.1 template<typename _IntType = int> result_type std::binomial_distribution< _IntType >::max ( ) const
[inline]
```

Returns the least upper bound value of the distribution.

Definition at line 3887 of file random.h.

```
4.513.3.2 template<typename _IntType = int> result_type std::binomial_distribution< _IntType >::min ( ) const
[inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 3880 of file random.h.

```
4.513.3.3 template<typename _IntType = int> template<typename _UniformRandomNumberGenerator> result_type
std::binomial_distribution< _IntType >::operator() ( _UniformRandomNumberGenerator & __urng ) [inline]
```

Generating functions.

Definition at line 3895 of file random.h.

```
4.513.3.4 template<typename _IntType = int> double std::binomial_distribution< _IntType >::p ( ) const [inline]
```

Returns the distribution  $p$  parameter.

Definition at line 3858 of file random.h.

```
4.513.3.5 template<typename _IntType = int> param_type std::binomial_distribution< _IntType >::param ( ) const
[inline]
```

Returns the parameter set of the distribution.

Definition at line 3865 of file random.h.

```
4.513.3.6  template<typename _IntType = int> void std::binomial_distribution<_IntType>::param ( const param_type &  
          __param ) [inline]
```

Sets the parameter set of the distribution.



## Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 3873 of file random.h.

4.513.3.7 `template<typename _IntType = int> void std::binomial_distribution< _IntType >::reset ( ) [inline]`

Resets the distribution state.

Definition at line 3844 of file random.h.

References `std::normal_distribution< _RealType >::reset()`.

4.513.3.8 `template<typename _IntType = int> _IntType std::binomial_distribution< _IntType >::t ( ) const [inline]`

Returns the distribution `t` parameter.

Definition at line 3851 of file random.h.

## 4.513.4 Friends And Related Function Documentation

4.513.4.1 `template<typename _IntType = int> template<typename _IntType1 , typename _CharT , typename _Traits > std::basic_ostream< _CharT, _Traits>& operator<< ( std::basic_ostream< _CharT, _Traits > & __os, const std::binomial_distribution< _IntType1 > & __x ) [friend]`

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

## Parameters

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

## Returns

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

4.513.4.2 `template<typename _IntType = int> bool operator== ( const binomial_distribution< _IntType > & __d1, const binomial_distribution< _IntType > & __d2 ) [friend]`

Return true if two `binomial` distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3931 of file random.h.

4.513.4.3 `template<typename _IntType = int> template<typename _IntType1 , typename _CharT , typename _Traits > std::basic_istream< _CharT, _Traits>& operator>> ( std::basic_istream< _CharT, _Traits > & __is, std::binomial_distribution< _IntType1 > & __x ) [friend]`

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

## Parameters

<code>__is</code>	An input stream.
-------------------	------------------

<code>__x</code>	A <code>binomial_distribution</code> random number generator engine.
------------------	--

## Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

4.514 `std::binomial_distribution<_IntType>::param_type` Struct Reference

## Public Types

- typedef `binomial_distribution<_IntType>` **distribution\_type**

## Public Member Functions

- **param\_type** (`_IntType __t=_IntType(1)`, `double __p=0.5`)
- `double p () const`
- `_IntType t () const`

## Friends

- class **binomial\_distribution<\_IntType>**
- `bool operator== (const param\_type &__p1, const param\_type &__p2)`

## 4.514.1 Detailed Description

```
template<typename _IntType = int> struct std::binomial_distribution<_IntType>::param_type
```

Parameter type.

Definition at line 3786 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.515 `std::cauchy_distribution<_RealType>` Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef `_RealType` [result\\_type](#)

## Public Member Functions

- **cauchy\_distribution** (\_RealType \_\_a=\_RealType(0), \_RealType \_\_b=\_RealType(1))
- **cauchy\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator> void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator> void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator> void **\_\_generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- \_RealType **a** () const
- \_RealType **b** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator> [result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator> [result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

## Friends

- bool **operator==** (const [cauchy\\_distribution](#) &\_\_d1, const [cauchy\\_distribution](#) &\_\_d2)

## 4.515.1 Detailed Description

```
template<typename _RealType = double>class std::cauchy_distribution< _RealType >
```

A `cauchy_distribution` random number distribution.

The formula for the normal probability mass function is  $p(x|a,b) = (\pi b(1 + (\frac{x-a}{b})^2))^{-1}$

Definition at line 2929 of file `random.h`.

## 4.515.2 Member Typedef Documentation

4.515.2.1 `template<typename _RealType = double> typedef _RealType std::cauchy_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 2932 of file `random.h`.

## 4.515.3 Member Function Documentation

4.515.3.1 `template<typename _RealType = double> result_type std::cauchy_distribution< _RealType >::max ( ) const`  
`[inline]`

Returns the least upper bound value of the distribution.

Definition at line 3020 of file `random.h`.

References `std::max()`.

4.515.3.2 `template<typename _RealType = double> result_type std::cauchy_distribution<_RealType>::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 3013 of file random.h.

4.515.3.3 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator> result_type std::cauchy_distribution<_RealType>::operator() ( _UniformRandomNumberGenerator &__urng ) [inline]`

Generating functions.

Definition at line 3028 of file random.h.

4.515.3.4 `template<typename _RealType = double> param_type std::cauchy_distribution<_RealType>::param ( ) const [inline]`

Returns the parameter set of the distribution.

Definition at line 2998 of file random.h.

4.515.3.5 `template<typename _RealType = double> void std::cauchy_distribution<_RealType>::param ( const param_type &__param ) [inline]`

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 3006 of file random.h.

4.515.3.6 `template<typename _RealType = double> void std::cauchy_distribution<_RealType>::reset ( ) [inline]`

Resets the distribution state.

Definition at line 2980 of file random.h.

#### 4.515.4 Friends And Related Function Documentation

4.515.4.1 `template<typename _RealType = double> bool operator== ( const cauchy_distribution<_RealType> &__d1, const cauchy_distribution<_RealType> &__d2 ) [friend]`

Return true if two Cauchy distributions have the same parameters.

Definition at line 3063 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

## 4.516 std::cauchy\_distribution<\_RealType>::param\_type Struct Reference

Public Types

- typedef [cauchy\\_distribution<\\_RealType>](#) **distribution\_type**

## Public Member Functions

- **param\_type** (\_RealType \_\_a=\_RealType(0), \_RealType \_\_b=\_RealType(1))
- \_RealType **a** () const
- \_RealType **b** () const

## Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

## 4.516.1 Detailed Description

template<typename \_RealType = double>struct std::cauchy\_distribution< \_RealType >::param\_type

Parameter type.

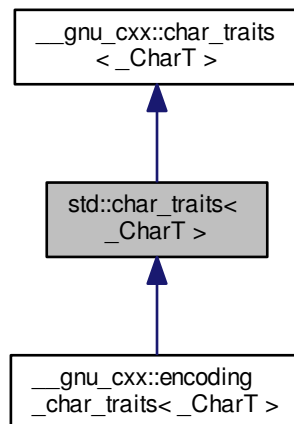
Definition at line 2938 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 4.517 std::char\_traits&lt; \_CharT &gt; Struct Template Reference

Inheritance diagram for std::char\_traits< \_CharT >:



## Public Types

- typedef \_CharT **char\_type**
- typedef \_Char\_types< \_CharT >::int\_type **int\_type**

- `typedef _Char_types< _CharT >::off_type off_type`
- `typedef _Char_types< _CharT >::pos_type pos_type`
- `typedef _Char_types< _CharT >::state_type state_type`

#### Static Public Member Functions

- static void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2)
- static char\_type \* **assign** (char\_type \* \_\_s, std::size\_t \_\_n, char\_type \_\_a)
- static int **compare** (const char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static char\_type \* **copy** (char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **eof** ()
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static const char\_type \* **find** (const char\_type \* \_\_s, std::size\_t \_\_n, const char\_type &\_\_a)
- static std::size\_t **length** (const char\_type \* \_\_s)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static char\_type \* **move** (char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c)
- static constexpr char\_type **to\_char\_type** (const int\_type &\_\_c)
- static constexpr int\_type **to\_int\_type** (const char\_type &\_\_c)

#### 4.517.1 Detailed Description

`template<class _CharT>struct std::char_traits< _CharT >`

Basis for explicit traits specializations.

#### Note

For any given actual character type, this definition is probably wrong. Since this is just a thin wrapper around `__gnu_cxx::char_traits`, it is possible to achieve a more appropriate definition by specializing `__gnu_cxx::char_traits`.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt05ch13s03.html> for advice on how to make use of this class for *unusual* character types. Also, check out `include/ext/pod_char_traits.h`.

Definition at line 227 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 4.518 `std::char_traits< __gnu_cxx::character< V, I, S > >` Struct Template Reference

#### Public Types

- `typedef __gnu_cxx::character< V, I, S > char_type`
- `typedef char_type::int_type int_type`
- `typedef streamoff off_type`
- `typedef fpos< state_type > pos_type`
- `typedef char_type::state_type state_type`

### Static Public Member Functions

- static void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2)
- static char\_type \* **assign** (char\_type \*\_\_s, size\_t \_\_n, char\_type \_\_a)
- static int **compare** (const char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static char\_type \* **copy** (char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static int\_type **eof** ()
- static bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static const char\_type \* **find** (const char\_type \*\_\_s, size\_t \_\_n, const char\_type &\_\_a)
- static size\_t **length** (const char\_type \*\_\_s)
- static bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static char\_type \* **move** (char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static int\_type **not\_eof** (const int\_type &\_\_c)
- static char\_type **to\_char\_type** (const int\_type &\_\_i)
- static int\_type **to\_int\_type** (const char\_type &\_\_c)

#### 4.518.1 Detailed Description

template<typename V, typename I, typename S>struct std::char\_traits< \_\_gnu\_cxx::character< V, I, S > >

char\_traits<\_\_gnu\_cxx::character> specialization.

Definition at line 95 of file pod\_char\_traits.h.

The documentation for this struct was generated from the following file:

- [pod\\_char\\_traits.h](#)

### 4.519 std::char\_traits< char > Struct Template Reference

#### Public Types

- typedef char **char\_type**
- typedef int **int\_type**
- typedef streamoff **off\_type**
- typedef streampos **pos\_type**
- typedef mbstate\_t **state\_type**

#### Static Public Member Functions

- static void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2) noexcept
- static char\_type \* **assign** (char\_type \*\_\_s, size\_t \_\_n, char\_type \_\_a)
- static int **compare** (const char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static char\_type \* **copy** (char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static constexpr int\_type **eof** () noexcept
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2) noexcept
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2) noexcept
- static const char\_type \* **find** (const char\_type \*\_\_s, size\_t \_\_n, const char\_type &\_\_a)
- static size\_t **length** (const char\_type \*\_\_s)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2) noexcept

- static `char_type * move` (`char_type *__s1`, `const char_type *__s2`, `size_t __n`)
- static constexpr `int_type not_eof` (`const int_type &__c`) noexcept
- static constexpr `char_type to_char_type` (`const int_type &__c`) noexcept
- static constexpr `int_type to_int_type` (`const char_type &__c`) noexcept

#### 4.519.1 Detailed Description

`template<> struct std::char_traits< char >`

##### 21.1.3.1 `char_traits` specializations

Definition at line 233 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 4.520 `std::char_traits< wchar_t >` Struct Template Reference

### Public Types

- typedef `wchar_t char_type`
- typedef `wint_t int_type`
- typedef [streamoff](#) `off_type`
- typedef [wstreampos](#) `pos_type`
- typedef `mbstate_t state_type`

### Static Public Member Functions

- static void **assign** (`char_type &__c1`, `const char_type &__c2`) noexcept
- static `char_type * assign` (`char_type *__s`, `size_t __n`, `char_type __a`)
- static `int compare` (`const char_type *__s1`, `const char_type *__s2`, `size_t __n`)
- static `char_type * copy` (`char_type *__s1`, `const char_type *__s2`, `size_t __n`)
- static constexpr `int_type eof` () noexcept
- static constexpr `bool eq` (`const char_type &__c1`, `const char_type &__c2`) noexcept
- static constexpr `bool eq_int_type` (`const int_type &__c1`, `const int_type &__c2`) noexcept
- static `const char_type * find` (`const char_type *__s`, `size_t __n`, `const char_type &__a`)
- static `size_t length` (`const char_type *__s`)
- static constexpr `bool lt` (`const char_type &__c1`, `const char_type &__c2`) noexcept
- static `char_type * move` (`char_type *__s1`, `const char_type *__s2`, `size_t __n`)
- static constexpr `int_type not_eof` (`const int_type &__c`) noexcept
- static constexpr `char_type to_char_type` (`const int_type &__c`) noexcept
- static constexpr `int_type to_int_type` (`const char_type &__c`) noexcept

#### 4.520.1 Detailed Description

`template<> struct std::char_traits< wchar_t >`

##### 21.1.3.2 `char_traits` specializations

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

The documentation for this struct was generated from the following file:



- [char\\_traits.h](#)

## 4.521 `std::chi_squared_distribution<_RealType>` Class Template Reference

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

### Public Member Functions

- **`chi_squared_distribution`** (`_RealType __n=_RealType(1)`)
- **`chi_squared_distribution`** (const [param\\_type](#) &\_\_p)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` > void **`__generate`** (`_ForwardIterator __f`, `_ForwardIterator __t`, `_UniformRandomNumberGenerator &__urng`)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` > void **`__generate`** (`_ForwardIterator __f`, `_ForwardIterator __t`, `_UniformRandomNumberGenerator &__urng`, const [param\\_type](#) &\_\_p)
- template<typename `_UniformRandomNumberGenerator` > void **`__generate`** ([result\\_type](#) \* \_\_f, [result\\_type](#) \* \_\_t, `_UniformRandomNumberGenerator &__urng`)
- template<typename `_UniformRandomNumberGenerator` > void **`__generate`** ([result\\_type](#) \* \_\_f, [result\\_type](#) \* \_\_t, `_UniformRandomNumberGenerator &__urng`, const [param\\_type](#) &\_\_p)
- [result\\_type](#) **`max`** () const
- [result\\_type](#) **`min`** () const
- `_RealType` **`n`** () const
- template<typename `_UniformRandomNumberGenerator` > [result\\_type](#) **`operator()`** (`_UniformRandomNumberGenerator &__urng`)
- template<typename `_UniformRandomNumberGenerator` > [result\\_type](#) **`operator()`** (`_UniformRandomNumberGenerator &__urng`, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **`param`** () const
- void **`param`** (const [param\\_type](#) &\_\_param)
- void **`reset`** ()

### Friends

- template<typename `_RealType1` , typename `_CharT` , typename `_Traits` > `std::basic_ostream<_CharT, _Traits>` & **`operator<<`** (`std::basic_ostream<_CharT, _Traits> &__os`, const [std::chi\\_squared\\_distribution<\\_RealType1>](#) &\_\_x)
- bool **`operator==`** (const [chi\\_squared\\_distribution](#) &\_\_d1, const [chi\\_squared\\_distribution](#) &\_\_d2)
- template<typename `_RealType1` , typename `_CharT` , typename `_Traits` > `std::basic_istream<_CharT, _Traits>` & **`operator>>`** (`std::basic_istream<_CharT, _Traits> &__is`, [std::chi\\_squared\\_distribution<\\_RealType1>](#) &\_\_x)

## 4.521.1 Detailed Description

```
template<typename _RealType = double>class std::chi_squared_distribution< _RealType >
```

A chi\_squared\_distribution random number distribution.

The formula for the normal probability mass function is  $p(x|n) = \frac{x^{(n/2)-1}e^{-x/2}}{\Gamma(n/2)2^{n/2}}$

Definition at line 2719 of file random.h.

## 4.521.2 Member Typedef Documentation

```
4.521.2.1 template<typename _RealType = double> typedef _RealType std::chi_squared_distribution< _RealType
>::result_type
```

The type of the range of the distribution.

Definition at line 2722 of file random.h.

## 4.521.3 Member Function Documentation

```
4.521.3.1 template<typename _RealType = double> result_type std::chi_squared_distribution< _RealType >::max ( )
const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 2799 of file random.h.

References std::max().

```
4.521.3.2 template<typename _RealType = double> result_type std::chi_squared_distribution< _RealType >::min ( )
const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 2792 of file random.h.

```
4.521.3.3 template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type
std::chi_squared_distribution< _RealType >::operator() ( _UniformRandomNumberGenerator & __urng )
[inline]
```

Generating functions.

Definition at line 2807 of file random.h.

```
4.521.3.4 template<typename _RealType = double> param_type std::chi_squared_distribution< _RealType >::param ( )
const [inline]
```

Returns the parameter set of the distribution.

Definition at line 2777 of file random.h.

```
4.521.3.5 template<typename _RealType = double> void std::chi_squared_distribution< _RealType >::param ( const
param_type & __param ) [inline]
```

Sets the parameter set of the distribution.

## Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 2785 of file random.h.

4.521.3.6 `template<typename _RealType = double> void std::chi_squared_distribution< _RealType >::reset ( )`  
`[inline]`

Resets the distribution state.

Definition at line 2763 of file random.h.

References `std::gamma_distribution< _RealType >::reset()`.

## 4.521.4 Friends And Related Function Documentation

4.521.4.1 `template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > std::basic_ostream< _CharT, _Traits>& operator<< ( std::basic_ostream< _CharT, _Traits > & __os, const std::chi_squared_distribution< _RealType1 > & __x )` `[friend]`

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

## Parameters

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

## Returns

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

4.521.4.2 `template<typename _RealType = double> bool operator==( const chi_squared_distribution< _RealType > & __d1, const chi_squared_distribution< _RealType > & __d2 )` `[friend]`

Return true if two Chi-squared distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2858 of file random.h.

4.521.4.3 `template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > std::basic_istream< _CharT, _Traits>& operator>> ( std::basic_istream< _CharT, _Traits > & __is, std::chi_squared_distribution< _RealType1 > & __x )` `[friend]`

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

## Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>chi_squared_distribution</code> random number generator engine.

## Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

4.522 `std::chi_squared_distribution<_RealType>::param_type` Struct Reference

## Public Types

- typedef `chi_squared_distribution<_RealType>` **distribution\_type**

## Public Member Functions

- **param\_type** (`_RealType __n=_RealType(1)`)
- `_RealType n` () const

## Friends

- bool **operator==** (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

## 4.522.1 Detailed Description

```
template<typename _RealType = double>struct std::chi_squared_distribution<_RealType>::param_type
```

Parameter type.

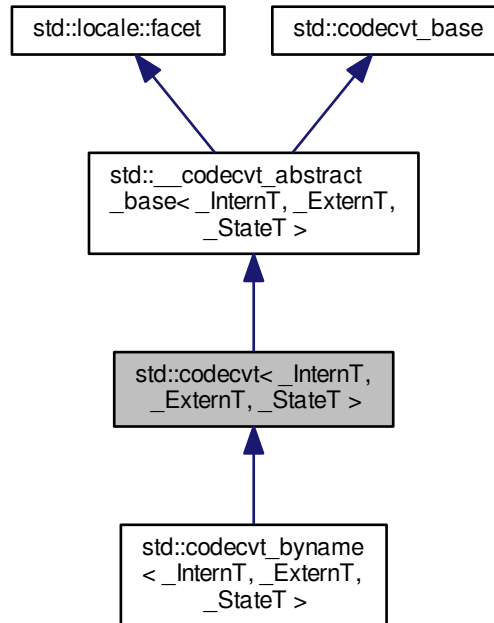
Definition at line 2728 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

#### 4.523 `std::codecvt<_InternT, _ExternT, _StateT >` Class Template Reference

Inheritance diagram for `std::codecvt<_InternT, _ExternT, _StateT >`:



##### Public Types

- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state\_type**

##### Public Member Functions

- **codecvt** (`size_t __refs=0`)
- **codecvt** (`_c_locale __cloc, size_t __refs=0`)
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (`state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *__&__from_next, intern_type *__to, intern_type *__to_end, intern_type *__&__to_next`) const
- int **length** (`state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max`) const
- int **max\_length** () const throw ()
- result **out** (`state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *__&__from_next, extern_type *__to, extern_type *__to_end, extern_type *__&__to_next`) const
- result **unshift** (`state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *__&__to_next`) const

## Static Public Attributes

- static [locale::id](#) id

## Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_codecvt**

## 4.523.1 Detailed Description

template<typename \_InternT, typename \_ExternT, typename \_StateT>class std::codecvt<\_InternT, \_ExternT, \_StateT >

Primary class template codecvt.

NB: Generic, mostly useless implementation.

Definition at line 276 of file codecvt.h.

## 4.523.2 Member Function Documentation

4.523.2.1 template<typename \_InternT, typename \_ExternT, typename \_StateT> virtual result std::codecvt<\_InternT, \_ExternT, \_StateT>::do\_out ( state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next ) const [protected], [virtual]

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, \\_StateT >](#).

```
4.523.2.2 template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base<
    _InternT, _ExternT, _StateT >::in ( state_type & __state, const extern_type * __from, const extern_type * __from_end,
    const extern_type * & __from_next, intern_type * __to, intern_type * __to_end, intern_type * & __to_next ) const
    [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

#### Returns

`codecvt_base::result`.

Definition at line 196 of file `codecvt.h`.

```
4.523.2.3 template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base<
    _InternT, _ExternT, _StateT >::out ( state_type & __state, const intern_type * __from, const intern_type * __from_end,
    const intern_type * & __from_next, extern_type * __to, extern_type * __to_end, extern_type * & __to_next ) const
    [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

#### Returns

`codecvt_base::result`.

Definition at line 116 of file `codecvt.h`.

```
4.523.2.4 template<typename _InternT, typename _ExternT, typename _StateT> result std:: __codecvt_abstract_base<
    _InternT, _ExternT, _StateT>::unshift ( state_type & __state, extern_type * __to, extern_type * __to_end, extern_type
    *& __to_next ) const    [inline],[inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

#### Parameters

<code>__state</code>	Persistent conversion state data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

#### Returns

`codecvt_base::result`.

Definition at line 155 of file `codecvt.h`.

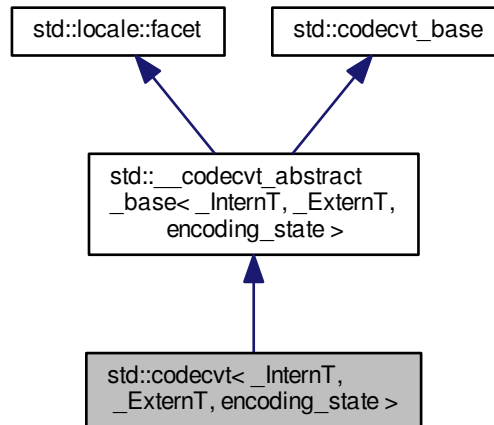
The documentation for this class was generated from the following file:

- [codecvt.h](#)



#### 4.524 `std::codecvt< _InternT, _ExternT, encoding_state >` Class Template Reference

Inheritance diagram for `std::codecvt< _InternT, _ExternT, encoding_state >`:



##### Public Types

- typedef `state_type::descriptor_type` **descriptor\_type**
- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `__gnu_cxx::encoding_state` **state\_type**

##### Public Member Functions

- **codecvt** (`size_t __refs=0`)
- **codecvt** (`state_type &__enc, size_t __refs=0`)
- **bool always\_noconv** () const throw ()
- **int encoding** () const throw ()
- **result in** (`state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *__&__from_next, intern_type *__to, intern_type *__to_end, intern_type *__&__to_next`) const
- **int length** (`state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max`) const
- **int max\_length** () const throw ()
- **result out** (`state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *__&__from_next, extern_type *__to, extern_type *__to_end, extern_type *__&__to_next`) const
- **result unshift** (`state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *__&__to_next`) const

##### Static Public Attributes

- static `locale::id` **id**

## Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## 4.524.1 Detailed Description

template<typename \_InternT, typename \_ExternT>class std::codecvt<\_InternT, \_ExternT, encoding\_state >

codecvt<InternT, \_ExternT, encoding\_state> specialization.

Definition at line 230 of file codecvt\_specializations.h.

## 4.524.2 Member Function Documentation

4.524.2.1 template<typename \_InternT, typename \_ExternT > codecvt\_base::result std::codecvt<\_InternT, \_ExternT, encoding\_state >::do\_out ( state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next ) const [protected], [virtual]

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements std::\_\_codecvt\_abstract\_base<\_InternT, \_ExternT, encoding\_state >.

Definition at line 306 of file codecvt\_specializations.h.

4.524.2.2 **result** **std::\_\_codecvt\_abstract\_base**< **\_InternT**, **\_ExternT**, **encoding\_state** >::**in** ( **state\_type** & **\_\_state**, **const extern\_type** \* **\_\_from**, **const extern\_type** \* **\_\_from\_end**, **const extern\_type** \* & **\_\_from\_next**, **intern\_type** \* **\_\_to**, **intern\_type** \* **\_\_to\_end**, **intern\_type** \* & **\_\_to\_next** ) **const** [inline],[inherited]

Convert from external to internal character set.

Converts input string of **extern\_type** to output string of **intern\_type**. This is analogous to **mbsrtowcs**. It does this by calling **codecvt::do\_in**.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [**from**,**from\_end**) are converted and written to [**to**,**to\_end**). **from\_next** and **to\_next** are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, **from\_next** and **to\_next** are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of **codecvt\_base::result**. If all the input is converted, returns **codecvt\_base::ok**. If no conversion is necessary, returns **codecvt\_base::noconv**. If the input ends early or there is insufficient space in the output, returns **codecvt\_base::partial**. Otherwise the conversion failed and **codecvt\_base::error** is returned.

#### Parameters

<b>__state</b>	Persistent conversion state data.
<b>__from</b>	Start of input.
<b>__from_end</b>	End of input.
<b>__from_next</b>	Returns start of unconverted data.
<b>__to</b>	Start of output buffer.
<b>__to_end</b>	End of output buffer.
<b>__to_next</b>	Returns start of unused output area.

#### Returns

**codecvt\_base::result**.

Definition at line 196 of file **codecvt.h**.

4.524.2.3 **result** **std::\_\_codecvt\_abstract\_base**< **\_InternT**, **\_ExternT**, **encoding\_state** >::**out** ( **state\_type** & **\_\_state**, **const intern\_type** \* **\_\_from**, **const intern\_type** \* **\_\_from\_end**, **const intern\_type** \* & **\_\_from\_next**, **extern\_type** \* **\_\_to**, **extern\_type** \* **\_\_to\_end**, **extern\_type** \* & **\_\_to\_next** ) **const** [inline],[inherited]

Convert from internal to external character set.

Converts input string of **intern\_type** to output string of **extern\_type**. This is analogous to **wcsrtombs**. It does this by calling **codecvt::do\_out**.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [**from**,**from\_end**) are converted and written to [**to**,**to\_end**). **from\_next** and **to\_next** are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, **from\_next** and **to\_next** are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of **codecvt\_base::result**. If all the input is converted, returns **codecvt\_base::ok**. If no conversion is necessary, returns **codecvt\_base::noconv**. If the input ends early or there is insufficient space in the output, returns **codecvt\_base::partial**. Otherwise the conversion failed and **codecvt\_base::error** is returned.

## Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

## Returns

codecvt\_base::result.

Definition at line 116 of file codecvt.h.

References `std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >::do_out()`.

**4.524.2.4** `result std::__codecvt_abstract_base<_InternT, _ExternT, encoding_state >::unshift ( state_type & __state, extern_type * __to, extern_type * __to_end, extern_type *& __to_next ) const` [inline],[inherited]

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

## Parameters

<code>__state</code>	Persistent conversion state data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

## Returns

codecvt\_base::result.

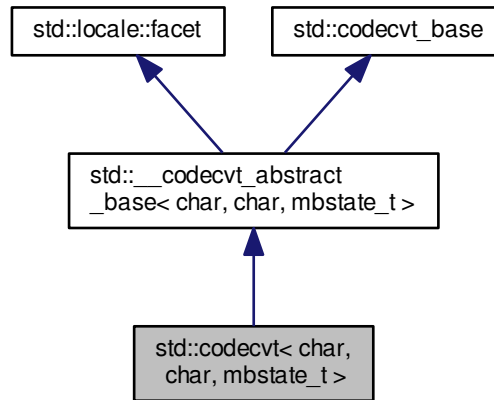
Definition at line 155 of file codecvt.h.

The documentation for this class was generated from the following file:

- [codecvt\\_specializations.h](#)

#### 4.525 `std::codecvt< char, char, mbstate_t >` Class Template Reference

Inheritance diagram for `std::codecvt< char, char, mbstate_t >`:



##### Public Types

- typedef char **extern\_type**
- typedef char **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `mbstate_t` **state\_type**

##### Public Member Functions

- **codecvt** (size\_t \_\_refs=0)
- **codecvt** (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

##### Static Public Attributes

- static `locale::id` **id**

## Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_type\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_codecvt**

## 4.525.1 Detailed Description

template<>class std::codecvt< char, char, mbstate\_t >

class codecvt<char, char, mbstate\_t> specialization.

Definition at line 340 of file codecvt.h.

## 4.525.2 Member Function Documentation

4.525.2.1 virtual result std::codecvt< char, char, mbstate\_t >::do\_out ( state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next ) const [protected], [virtual]

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

## See also

out for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base< char, char, mbstate\\_t >](#).

4.525.2.2 **result** **std::\_\_codecvt\_abstract\_base**< char , char , mbstate\_t >::in ( state\_type & \_\_state, const extern\_type \* \_\_from, const extern\_type \* \_\_from\_end, const extern\_type \* & \_\_from\_next, intern\_type \* \_\_to, intern\_type \* \_\_to\_end, intern\_type \* & \_\_to\_next ) const [inline], [inherited]

Convert from external to internal character set.

Converts input string of extern\_type to output string of intern\_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do\_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt\_base::result. If all the input is converted, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt\_base::partial. Otherwise the conversion failed and codecvt\_base::error is returned.

#### Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

#### Returns

codecvt\_base::result.

Definition at line 196 of file codecvt.h.

4.525.2.3 **result** **std::\_\_codecvt\_abstract\_base**< char , char , mbstate\_t >::out ( state\_type & \_\_state, const intern\_type \* \_\_from, const intern\_type \* \_\_from\_end, const intern\_type \* & \_\_from\_next, extern\_type \* \_\_to, extern\_type \* \_\_to\_end, extern\_type \* & \_\_to\_next ) const [inline], [inherited]

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This is analogous to wcsrtombs. It does this by calling codecvt::do\_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt\_base::result. If all the input is converted, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt\_base::partial. Otherwise the conversion failed and codecvt\_base::error is returned.

## Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

## Returns

`codecvt_base::result`.

Definition at line 116 of file `codecvt.h`.

**4.525.2.4** `result std::__codecvt_abstract_base< char , char , mbstate_t >::unshift ( state_type & __state, extern_type * __to, extern_type * __to_end, extern_type *& __to_next ) const` `[inline],[inherited]`

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

## Parameters

<code>__state</code>	Persistent conversion state data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.



**Returns**

codecvt\_base::result.

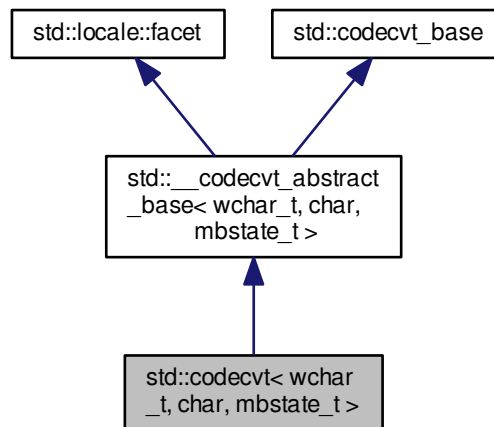
Definition at line 155 of file codecvt.h.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

#### 4.526 std::codecvt< wchar\_t, char, mbstate\_t > Class Template Reference

Inheritance diagram for std::codecvt< wchar\_t, char, mbstate\_t >:

**Public Types**

- typedef char **extern\_type**
- typedef wchar\_t **intern\_type**
- typedef codecvt\_base::result **result**
- typedef mbstate\_t **state\_type**

**Public Member Functions**

- **codecvt** (size\_t \_\_refs=0)
- **codecvt** (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const

- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const

#### Static Public Attributes

- static **locale::id** id

#### Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const

#### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

#### Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_codecvt**

#### 4.526.1 Detailed Description

template<>class std::codecvt< wchar\_t, char, mbstate\_t >

class codecvt<wchar\_t, char, mbstate\_t> specialization.

Definition at line 398 of file codecvt.h.

#### 4.526.2 Member Function Documentation

- 4.526.2.1 virtual result std::codecvt< wchar\_t, char, mbstate\_t >::do\_out ( state\_type & \_\_state, const intern\_type \* \_\_from, const intern\_type \* \_\_from\_end, const intern\_type \*& \_\_from\_next, extern\_type \* \_\_to, extern\_type \* \_\_to\_end, extern\_type \*& \_\_to\_next ) const [protected], [virtual]

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements `std::__codecvt_abstract_base< wchar_t, char, mbstate_t >`.

4.526.2.2 `result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::in ( state_type & __state, const extern_type * __from, const extern_type * __from_end, const extern_type * & __from_next, intern_type * __to, intern_type * __to_end, intern_type * & __to_next ) const` [inline], [inherited]

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

Returns

`codecvt_base::result`.

Definition at line 196 of file `codecvt.h`.

4.526.2.3 `result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::out ( state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type * & __from_next, extern_type * __to, extern_type * __to_end, extern_type * & __to_next ) const` [inline], [inherited]

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

#### Returns

`codecvt_base::result`.

Definition at line 116 of file `codecvt.h`.

**4.526.2.4** `result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::unshift ( state_type & __state, extern_type * __to, extern_type * __to_end, extern_type * & __to_next ) const` [inline], [inherited]

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

#### Parameters

<code>__state</code>	Persistent conversion state data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

#### Returns

`codecvt_base::result`.

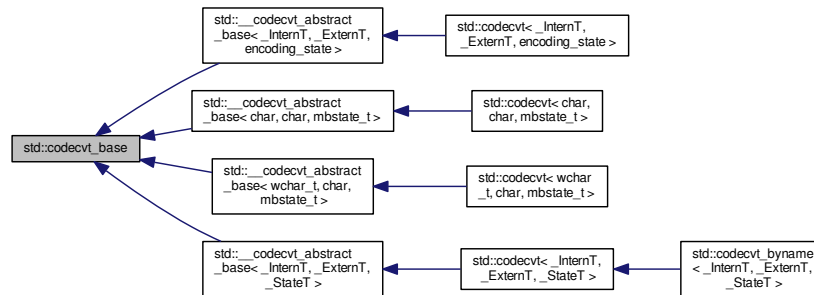
Definition at line 155 of file `codecvt.h`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 4.527 std::codecvt\_base Class Reference

Inheritance diagram for std::codecvt\_base:



### Public Types

- enum **result** { **ok**, **partial**, **error**, **noconv** }

### 4.527.1 Detailed Description

Empty base class for codecvt facet [22.2.1.5].

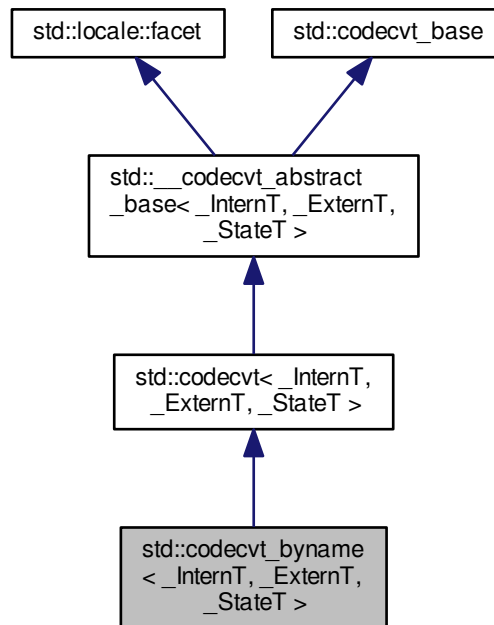
Definition at line 46 of file codecvt.h.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 4.528 std::codecvt\_byname&lt; \_InternT, \_ExternT, \_StateT &gt; Class Template Reference

Inheritance diagram for std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >:



## Public Types

- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state\_type**

## Public Member Functions

- **codecvt\_byname** (const char \*\_\_s, size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

## Static Public Attributes

- static `locale::id` `id`

## Protected Member Functions

- virtual bool `do_always_noconv` () const throw ()
- virtual int `do_encoding` () const throw ()
- virtual result `do_in` (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- virtual int `do_length` (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int `do_max_length` () const throw ()
- virtual result `do_out` (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- virtual result `do_unshift` (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const

## Static Protected Member Functions

- static \_\_c\_locale `_S_clone_c_locale` (\_\_c\_locale &\_\_cloc) throw ()
- static void `_S_create_c_locale` (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void `_S_destroy_c_locale` (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale `_S_get_c_locale` ()
- static const char \* `_S_get_c_name` () throw ()
- static \_\_c\_locale `_S_lc_ctype_c_locale` (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## Protected Attributes

- \_\_c\_locale `_M_c_locale_codecvt`

### 4.528.1 Detailed Description

```
template<typename _InternT, typename _ExternT, typename _StateT> class std::codecvt_byname< _InternT, _ExternT, _StateT >
```

class codecvt\_byname [22.2.1.6].

Definition at line 458 of file codecvt.h.

### 4.528.2 Member Function Documentation

4.528.2.1 `template<typename _InternT, typename _ExternT, typename _StateT> virtual result std::codecvt< _InternT, _ExternT, _StateT >::do_out ( state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next ) const` [protected], [virtual], [inherited]

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base< \\_InternT, \\_ExternT, \\_StateT >](#).

4.528.2.2 `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::in ( state_type & __state, const extern_type * __from, const extern_type * __from_end, const extern_type * & __from_next, intern_type * __to, intern_type * __to_end, intern_type * & __to_next ) const [inline], [inherited]`

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

#### Returns

`codecvt_base::result`.

Definition at line 196 of file `codecvt.h`.

4.528.2.3 `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::out ( state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type * & __from_next, extern_type * __to, extern_type * __to_end, extern_type * & __to_next ) const [inline], [inherited]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.



The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

#### Returns

`codecvt_base::result`.

Definition at line 116 of file `codecvt.h`.

```
4.528.2.4 template<typename _InternT, typename _ExternT, typename _StateT> result std:: __codecvt_abstract_base<
    _InternT, _ExternT, _StateT>::unshift ( state_type & __state, extern_type * __to, extern_type * __to_end, extern_type
    *& __to_next ) const    [inline],[inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

#### Parameters

<code>__state</code>	Persistent conversion state data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

#### Returns

`codecvt_base::result`.

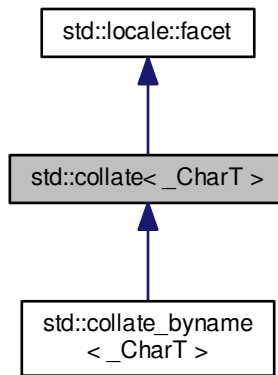
Definition at line 155 of file `codecvt.h`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

4.529 `std::collate<_CharT>` Class Template Reference

Inheritance diagram for `std::collate<_CharT>`:



## Public Types

- typedef `_CharT` `char_type`
- typedef `basic_string<_CharT>` `string_type`

## Public Member Functions

- `collate` (`size_t __refs=0`)
- `collate` (`__c_locale __cloc, size_t __refs=0`)
- `int _M_compare` (`const _CharT *, const _CharT *`) `const throw ()`
- `template<> int _M_compare` (`const char *, const char *`) `const throw()`
- `template<> int _M_compare` (`const wchar_t *, const wchar_t *`) `const throw()`
- `size_t _M_transform` (`_CharT *, const _CharT *, size_t`) `const throw ()`
- `template<> size_t _M_transform` (`char *, const char *, size_t`) `const throw()`
- `template<> size_t _M_transform` (`wchar_t *, const wchar_t *, size_t`) `const throw()`
- `int compare` (`const _CharT * __lo1, const _CharT * __hi1, const _CharT * __lo2, const _CharT * __hi2`) `const`
- `long hash` (`const _CharT * __lo, const _CharT * __hi`) `const`
- `string_type transform` (`const _CharT * __lo, const _CharT * __hi`) `const`

## Static Public Attributes

- static `locale::id` `id`

### Protected Member Functions

- virtual `~collate()`
- virtual `int do_compare(const _CharT * __lo1, const _CharT * __hi1, const _CharT * __lo2, const _CharT * __hi2) const`
- virtual `long do_hash(const _CharT * __lo, const _CharT * __hi) const`
- virtual `string_type do_transform(const _CharT * __lo, const _CharT * __hi) const`

### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale(__c_locale & __cloc) throw()`
- static `void _S_create_c_locale(__c_locale & __cloc, const char * __s, __c_locale __old=0)`
- static `void _S_destroy_c_locale(__c_locale & __cloc)`
- static `__c_locale _S_get_c_locale()`
- static `const char * _S_get_c_name() throw()`
- static `__c_locale _S_lc_type_c_locale(__c_locale __cloc, const char * __s)`

### Protected Attributes

- `__c_locale _M_c_locale_collate`

#### 4.529.1 Detailed Description

`template<typename _CharT> class std::collate<_CharT>`

Facet for localized string comparison.

This facet encapsulates the code to compare strings in a localized manner.

The collate template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the collate facet.

Definition at line 584 of file `locale_classes.h`.

#### 4.529.2 Member Typedef Documentation

4.529.2.1 `template<typename _CharT> typedef _CharT std::collate<_CharT>::char_type`

Public typedefs.

Definition at line 590 of file `locale_classes.h`.

4.529.2.2 `template<typename _CharT> typedef basic_string<_CharT> std::collate<_CharT>::string_type`

Public typedefs.

Definition at line 591 of file `locale_classes.h`.

#### 4.529.3 Constructor & Destructor Documentation

4.529.3.1 `template<typename _CharT> std::collate<_CharT>::collate(size_t __refs = 0) [inline], [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

## Parameters

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

Definition at line 611 of file `locale_classes.h`.

**4.529.3.2** `template<typename _CharT> std::collate<_CharT>::collate ( __c_locale __cloc, size_t __refs = 0 )`  
`[inline], [explicit]`

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

## Parameters

<code>__cloc</code>	The C locale.
<code>__refs</code>	Passed to the base facet class.

Definition at line 625 of file `locale_classes.h`.

**4.529.3.3** `template<typename _CharT> virtual std::collate<_CharT>::~~collate ( )` `[inline], [protected], [virtual]`

Destructor.

Definition at line 688 of file `locale_classes.h`.

**4.529.4 Member Function Documentation**

**4.529.4.1** `template<typename _CharT> int std::collate<_CharT>::compare ( const _CharT * __lo1, const _CharT * __hi1, const _CharT * __lo2, const _CharT * __hi2 ) const` `[inline]`

Compare two strings.

This function compares two strings and returns the result by calling `collate::do_compare()`.

## Parameters

<code>__lo1</code>	Start of string 1.
<code>__hi1</code>	End of string 1.
<code>__lo2</code>	Start of string 2.
<code>__hi2</code>	End of string 2.

## Returns

1 if `string1 > string2`, -1 if `string1 < string2`, else 0.

Definition at line 642 of file `locale_classes.h`.

**4.529.4.2** `template<typename _CharT> virtual int std::collate<_CharT>::do_compare ( const _CharT * __lo1, const _CharT * __hi1, const _CharT * __lo2, const _CharT * __hi2 ) const` `[protected], [virtual]`

Compare two strings.

This function is a hook for derived classes to change the value returned.

## See also

`compare()`.

## Parameters

<code>__lo1</code>	Start of string 1.
<code>__hi1</code>	End of string 1.
<code>__lo2</code>	Start of string 2.
<code>__hi2</code>	End of string 2.

## Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

4.529.4.3 `template<typename _CharT> virtual long std::collate<_CharT>::do_hash ( const _CharT * __lo, const _CharT * __hi ) const` [protected], [virtual]

Return hash of a string.

This function computes and returns a hash on the input string. This function is a hook for derived classes to change the value returned.

## Parameters

<code>__lo</code>	Start of string.
<code>__hi</code>	End of string.

## Returns

Hash value.

4.529.4.4 `template<typename _CharT> virtual string_type std::collate<_CharT>::do_transform ( const _CharT * __lo, const _CharT * __hi ) const` [protected], [virtual]

Transform string to comparable form.

This function is a hook for derived classes to change the value returned.

## Parameters

<code>__lo</code>	Start.
<code>__hi</code>	End.

## Returns

transformed string.

4.529.4.5 `template<typename _CharT> long std::collate<_CharT>::hash ( const _CharT * __lo, const _CharT * __hi ) const` [inline]

Return hash of a string.

This function computes and returns a hash on the input string. It does so by returning `collate::do_hash()`.

## Parameters

<code>__lo</code>	Start of string.
<code>__hi</code>	End of string.

**Returns**

Hash value.

Definition at line 675 of file `locale_classes.h`.

**4.529.4.6** `template<typename _CharT> string_type std::collate<_CharT>::transform ( const _CharT * __lo, const _CharT * __hi ) const [inline]`

Transform string to comparable form.

This function is a wrapper for `strxfrm` functionality. It takes the input string and returns a modified string that can be directly compared to other transformed strings. In the C locale, this function just returns a copy of the input string. In some other locales, it may replace two chars with one, change a char for another, etc. It does so by returning `collate::do_transform()`.

**Parameters**

<code>__lo</code>	Start of string.
<code>__hi</code>	End of string.

**Returns**

Transformed `string_type`.

Definition at line 661 of file `locale_classes.h`.

**4.529.5 Member Data Documentation**

**4.529.5.1** `template<typename _CharT> locale::id std::collate<_CharT>::id [static]`

Numpunct facet id.

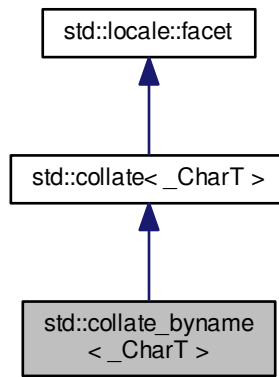
Definition at line 601 of file `locale_classes.h`.

The documentation for this class was generated from the following file:

- [locale\\_classes.h](#)

## 4.530 std::collate\_byname&lt;\_CharT&gt; Class Template Reference

Inheritance diagram for std::collate\_byname<\_CharT>:



## Public Types

- typedef `_CharT` [char\\_type](#)
- typedef [basic\\_string](#)<`_CharT`> [string\\_type](#)

## Public Member Functions

- **collate\_byname** (const char \*\_\_s, size\_t \_\_refs=0)
- int **\_M\_compare** (const \_CharT \*, const \_CharT \*) const throw ()
- template<> int **\_M\_compare** (const char \*, const char \*) const throw()
- template<> int **\_M\_compare** (const wchar\_t \*, const wchar\_t \*) const throw()
- size\_t **\_M\_transform** (\_CharT \*, const \_CharT \*, size\_t) const throw ()
- template<> size\_t **\_M\_transform** (char \*, const char \*, size\_t) const throw()
- template<> size\_t **\_M\_transform** (wchar\_t \*, const wchar\_t \*, size\_t) const throw()
- int [compare](#) (const \_CharT \*\_\_lo1, const \_CharT \*\_\_hi1, const \_CharT \*\_\_lo2, const \_CharT \*\_\_hi2) const
- long [hash](#) (const \_CharT \*\_\_lo, const \_CharT \*\_\_hi) const
- [string\\_type transform](#) (const \_CharT \*\_\_lo, const \_CharT \*\_\_hi) const

## Static Public Attributes

- static [locale::id](#) `id`

## Protected Member Functions

- virtual int [do\\_compare](#) (const \_CharT \*\_\_lo1, const \_CharT \*\_\_hi1, const \_CharT \*\_\_lo2, const \_CharT \*\_\_hi2) const
- virtual long [do\\_hash](#) (const \_CharT \*\_\_lo, const \_CharT \*\_\_hi) const
- virtual [string\\_type do\\_transform](#) (const \_CharT \*\_\_lo, const \_CharT \*\_\_hi) const



### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

### Protected Attributes

- `__c_locale _M_c_locale_collate`

#### 4.530.1 Detailed Description

`template<typename _CharT>class std::collate_byname<_CharT>`

class `collate_byname` [22.2.4.2].

Definition at line 758 of file `locale_classes.h`.

#### 4.530.2 Member Typedef Documentation

4.530.2.1 `template<typename _CharT> typedef _CharT std::collate_byname<_CharT>::char_type`

Public typedefs.

Definition at line 763 of file `locale_classes.h`.

4.530.2.2 `template<typename _CharT> typedef basic_string<_CharT> std::collate_byname<_CharT>::string_type`

Public typedefs.

Definition at line 764 of file `locale_classes.h`.

#### 4.530.3 Member Function Documentation

4.530.3.1 `template<typename _CharT> int std::collate<_CharT>::compare ( const _CharT * __lo1, const _CharT * __hi1, const _CharT * __lo2, const _CharT * __hi2 ) const` [inline], [inherited]

Compare two strings.

This function compares two strings and returns the result by calling `collate::do_compare()`.

##### Parameters

<code>__lo1</code>	Start of string 1.
<code>__hi1</code>	End of string 1.
<code>__lo2</code>	Start of string 2.
<code>__hi2</code>	End of string 2.

**Returns**

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 642 of file locale\_classes.h.

**4.530.3.2** `template<typename _CharT> virtual int std::collate<_CharT>::do_compare ( const _CharT * __lo1, const _CharT * __hi1, const _CharT * __lo2, const _CharT * __hi2 ) const` [protected], [virtual], [inherited]

Compare two strings.

This function is a hook for derived classes to change the value returned.

**See also**

compare().

**Parameters**

<code>__lo1</code>	Start of string 1.
<code>__hi1</code>	End of string 1.
<code>__lo2</code>	Start of string 2.
<code>__hi2</code>	End of string 2.

**Returns**

1 if string1 > string2, -1 if string1 < string2, else 0.

**4.530.3.3** `template<typename _CharT> virtual long std::collate<_CharT>::do_hash ( const _CharT * __lo, const _CharT * __hi ) const` [protected], [virtual], [inherited]

Return hash of a string.

This function computes and returns a hash on the input string. This function is a hook for derived classes to change the value returned.

**Parameters**

<code>__lo</code>	Start of string.
<code>__hi</code>	End of string.

**Returns**

Hash value.

**4.530.3.4** `template<typename _CharT> virtual string_type std::collate<_CharT>::do_transform ( const _CharT * __lo, const _CharT * __hi ) const` [protected], [virtual], [inherited]

Transform string to comparable form.

This function is a hook for derived classes to change the value returned.

**Parameters**

<code>__lo</code>	Start.
<code>__hi</code>	End.

**Returns**

transformed string.

**4.530.3.5** `template<typename _CharT> long std::collate<_CharT>::hash ( const _CharT * __lo, const _CharT * __hi ) const`  
`[inline], [inherited]`

Return hash of a string.

This function computes and returns a hash on the input string. It does so by returning `collate::do_hash()`.

**Parameters**

<code>__lo</code>	Start of string.
<code>__hi</code>	End of string.

**Returns**

Hash value.

Definition at line 675 of file `locale_classes.h`.

**4.530.3.6** `template<typename _CharT> string_type std::collate<_CharT>::transform ( const _CharT * __lo, const _CharT * __hi ) const`  
`[inline], [inherited]`

Transform string to comparable form.

This function is a wrapper for `strxfrm` functionality. It takes the input string and returns a modified string that can be directly compared to other transformed strings. In the C locale, this function just returns a copy of the input string. In some other locales, it may replace two chars with one, change a char for another, etc. It does so by returning `collate::do_transform()`.

**Parameters**

<code>__lo</code>	Start of string.
<code>__hi</code>	End of string.

**Returns**

Transformed `string_type`.

Definition at line 661 of file `locale_classes.h`.

**4.530.4 Member Data Documentation**

**4.530.4.1** `template<typename _CharT> locale::id std::collate<_CharT>::id` `[static], [inherited]`

Numpunct facet id.

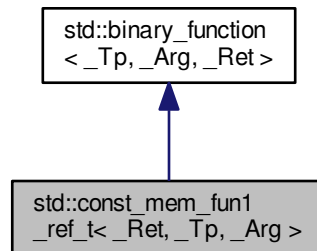
Definition at line 601 of file `locale_classes.h`.

The documentation for this class was generated from the following file:

- [locale\\_classes.h](#)

## 4.531 std::const\_mem\_fun1\_ref\_t&lt; \_Ret, \_Tp, \_Arg &gt; Class Template Reference

Inheritance diagram for std::const\_mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >:



## Public Types

- typedef \_Tp [first\\_argument\\_type](#)
- typedef \_Ret [result\\_type](#)
- typedef \_Arg [second\\_argument\\_type](#)

## Public Member Functions

- **const\_mem\_fun1\_ref\_t** (\_Ret(\_Tp::\*\_\_pf)(\_Arg) const)
- **\_Ret operator()** (const \_Tp &\_\_r, \_Arg \_\_x) const

## 4.531.1 Detailed Description

```
template<typename _Ret, typename _Tp, typename _Arg>class std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >
```

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

Definition at line 668 of file stl\_function.h.

## 4.531.2 Member Typedef Documentation

4.531.2.1 typedef \_Tp std::binary\_function< \_Tp, \_Arg, \_Ret >::[first\\_argument\\_type](#) [inherited]

[first\\_argument\\_type](#) is the type of the first argument

Definition at line 117 of file stl\_function.h.

4.531.2.2 typedef \_Ret std::binary\_function< \_Tp, \_Arg, \_Ret >::[result\\_type](#) [inherited]

[result\\_type](#) is the return type

Definition at line 123 of file stl\_function.h.

#### 4.531.2.3 `typedef _Arg std::binary_function< _Tp, _Arg, _Ret >::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

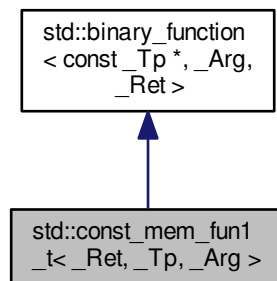
Definition at line 120 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

#### 4.532 `std::const_mem_fun1_t< _Ret, _Tp, _Arg >` Class Template Reference

Inheritance diagram for `std::const_mem_fun1_t< _Ret, _Tp, _Arg >`:



##### Public Types

- `typedef const _Tp *` [first\\_argument\\_type](#)
- `typedef _Ret` [result\\_type](#)
- `typedef _Arg` [second\\_argument\\_type](#)

##### Public Member Functions

- `const_mem_fun1_t ( _Ret(_Tp::* __pf)( _Arg) const)`
- `_Ret operator() (const _Tp * __p, _Arg __x) const`

#### 4.532.1 Detailed Description

`template<typename _Ret, typename _Tp, typename _Arg> class std::const_mem_fun1_t< _Ret, _Tp, _Arg >`

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

Definition at line 632 of file `stl_function.h`.

## 4.532.2 Member Typedef Documentation

## 4.532.2.1 typedef const\_Tp \* std::binary\_function&lt; const\_Tp \*, \_Arg, \_Ret &gt;::first\_argument\_type [inherited]

first\_argument\_type is the type of the first argument

Definition at line 117 of file stl\_function.h.

## 4.532.2.2 typedef \_Ret std::binary\_function&lt; const\_Tp \*, \_Arg, \_Ret &gt;::result\_type [inherited]

result\_type is the return type

Definition at line 123 of file stl\_function.h.

## 4.532.2.3 typedef \_Arg std::binary\_function&lt; const\_Tp \*, \_Arg, \_Ret &gt;::second\_argument\_type [inherited]

second\_argument\_type is the type of the second argument

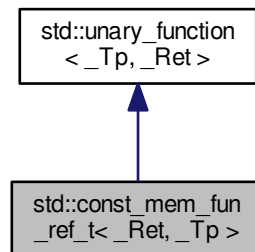
Definition at line 120 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 4.533 std::const\_mem\_fun\_ref\_t&lt; \_Ret, \_Tp &gt; Class Template Reference

Inheritance diagram for std::const\_mem\_fun\_ref\_t< \_Ret, \_Tp >:



## Public Types

- typedef \_Tp [argument\\_type](#)
- typedef \_Ret [result\\_type](#)

## Public Member Functions

- **const\_mem\_fun\_ref\_t** (\_Ret(\_Tp::\* \_\_pf)() const)
- **\_Ret operator()** (const \_Tp &\_\_r) const

#### 4.533.1 Detailed Description

`template<typename _Ret, typename _Tp>class std::const_mem_fun_ref_t< _Ret, _Tp >`

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

Definition at line 596 of file `stl_function.h`.

#### 4.533.2 Member Typedef Documentation

4.533.2.1 `typedef _Tp std::unary_function< _Tp, _Ret >::argument_type` `[inherited]`

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

4.533.2.2 `typedef _Ret std::unary_function< _Tp, _Ret >::result_type` `[inherited]`

`result_type` is the return type

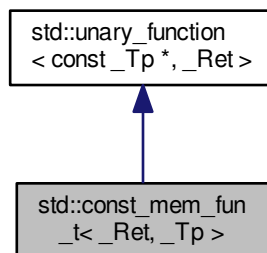
Definition at line 107 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

#### 4.534 `std::const_mem_fun_t< _Ret, _Tp >` Class Template Reference

Inheritance diagram for `std::const_mem_fun_t< _Ret, _Tp >`:



#### Public Types

- `typedef const_Tp *` [argument\\_type](#)
- `typedef _Ret` [result\\_type](#)

#### Public Member Functions

- `const_mem_fun_t(_Ret(_Tp::*__pf)() const)`

- `_Ret operator()` (`const _Tp *__p`) `const`

#### 4.534.1 Detailed Description

```
template<typename _Ret, typename _Tp>class std::const_mem_fun_t<_Ret, _Tp>
```

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

Definition at line 560 of file `stl_function.h`.

#### 4.534.2 Member Typedef Documentation

4.534.2.1 `typedef const_Tp * std::unary_function< const_Tp *, _Ret >::argument_type` `[inherited]`

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

4.534.2.2 `typedef _Ret std::unary_function< const_Tp *, _Ret >::result_type` `[inherited]`

`result_type` is the return type

Definition at line 107 of file `stl_function.h`.

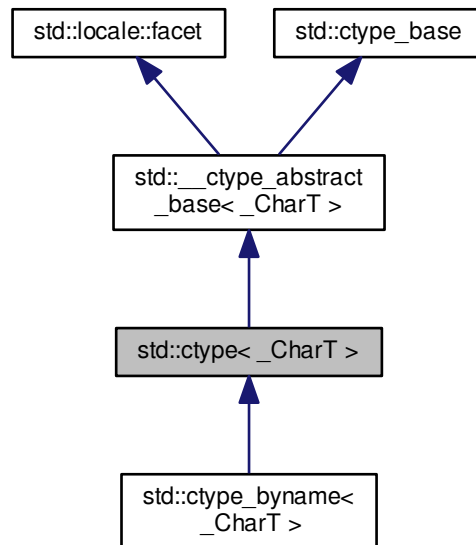
The documentation for this class was generated from the following file:

- [stl\\_function.h](#)



#### 4.535 `std::ctype<_CharT>` Class Template Reference

Inheritance diagram for `std::ctype<_CharT>`:



#### Public Types

- `typedef const int * __to_type`
- `typedef _CharT char_type`
- `typedef \_\_ctype\_abstract\_base<_CharT>::mask mask`

#### Public Member Functions

- `ctype` (`size_t __refs=0`)
- `bool is` (`mask __m, char_type __c`) `const`
- `const char_type * is` (`const char_type * __lo, const char_type * __hi, mask * __vec`) `const`
- `char narrow` (`char_type __c, char __dfault`) `const`
- `const char_type * narrow` (`const char_type * __lo, const char_type * __hi, char __dfault, char * __to`) `const`
- `const char_type * scan_is` (`mask __m, const char_type * __lo, const char_type * __hi`) `const`
- `const char_type * scan_not` (`mask __m, const char_type * __lo, const char_type * __hi`) `const`
- `char_type tolower` (`char_type __c`) `const`
- `const char_type * tolower` (`char_type * __lo, const char_type * __hi`) `const`
- `char_type toupper` (`char_type __c`) `const`
- `const char_type * toupper` (`char_type * __lo, const char_type * __hi`) `const`
- `char_type widen` (`char __c`) `const`
- `const char * widen` (`const char * __lo, const char * __hi, char_type * __to`) `const`

## Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

## Protected Member Functions

- virtual bool [do\\_is](#) (mask \_\_m, [char\\_type](#) \_\_c) const
- virtual const [char\\_type](#) \* [do\\_is](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, mask \* \_\_vec) const
- virtual [char](#) [do\\_narrow](#) ([char\\_type](#), [char](#) \_\_default) const
- virtual const [char\\_type](#) \* [do\\_narrow](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, [char](#) \_\_default, [char](#) \* \_\_to) const
- virtual const [char\\_type](#) \* [do\\_scan\\_is](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual const [char\\_type](#) \* [do\\_scan\\_not](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual [char\\_type](#) [do\\_tolower](#) ([char\\_type](#) \_\_c) const
- virtual const [char\\_type](#) \* [do\\_tolower](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual [char\\_type](#) [do\\_toupper](#) ([char\\_type](#) \_\_c) const
- virtual const [char\\_type](#) \* [do\\_toupper](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual [char\\_type](#) [do\\_widen](#) ([char](#) \_\_c) const
- virtual const [char](#) \* [do\\_widen](#) (const [char](#) \* \_\_lo, const [char](#) \* \_\_hi, [char\\_type](#) \* \_\_dest) const

## Static Protected Member Functions

- static [\\_\\_c\\_locale](#) [\\_S\\_clone\\_c\\_locale](#) ([\\_\\_c\\_locale](#) & \_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) ([\\_\\_c\\_locale](#) & \_\_cloc, const [char](#) \* \_\_s, [\\_\\_c\\_locale](#) \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) ([\\_\\_c\\_locale](#) & \_\_cloc)
- static [\\_\\_c\\_locale](#) [\\_S\\_get\\_c\\_locale](#) ()
- static const [char](#) \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static [\\_\\_c\\_locale](#) [\\_S\\_lc\\_ctype\\_c\\_locale](#) ([\\_\\_c\\_locale](#) \_\_cloc, const [char](#) \* \_\_s)

## 4.535.1 Detailed Description

```
template<typename _CharT>class std::ctype< _CharT >
```

Primary class template ctype facet.

This template class defines classification and conversion functions for character sets. It wraps ctype functionality. Ctype gets used by streams for many I/O operations.

This template provides the protected virtual functions the developer will have to replace in a derived class or specialization to make a working facet. The public functions that access them are defined in `__ctype_abstract_base`, to allow for

implementation flexibility. See `ctype<wchar_t>` for an example. The functions are documented in `__ctype_abstract_base`.

Note: implementations are provided for all the protected virtual functions, but will likely not be useful.

Definition at line 605 of file `locale_facets.h`.

#### 4.535.2 Member Function Documentation

**4.535.2.1** `template<typename _CharT> virtual bool std::ctype<_CharT>::do_is ( mask __m, char_type __c ) const`  
`[protected], [virtual]`

Test `char_type` classification.

This function finds a mask `M` for `c` and compares it to mask `m`.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

##### Parameters

<code>__c</code>	The <code>char_type</code> to find the mask of.
<code>__m</code>	The mask to compare against.

##### Returns

`(M & __m) != 0`.

Implements `std::__ctype_abstract_base<_CharT>`.

**4.535.2.2** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_is ( const char_type * __lo, const char_type * __hi, mask * __vec ) const`  
`[protected], [virtual]`

Return a mask array.

This function finds the mask for each `char_type` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the input.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

##### Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.

##### Returns

`__hi`.

Implements `std::__ctype_abstract_base<_CharT>`.

**4.535.2.3** `template<typename _CharT> virtual char std::ctype<_CharT>::do_narrow ( char_type __c, char __dfault ) const`  
`[protected], [virtual]`

Narrow `char_type` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

<code>__c</code>	The char_type to convert.
<code>__dfault</code>	Char to return if conversion fails.

#### Returns

The converted char.

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

Referenced by `std::ctype< char >::narrow()`.

**4.535.2.4** `template<typename _CharT> virtual const char_type* std::ctype< _CharT >::do_narrow ( const char_type *  
__lo, const char_type * __hi, char __dfault, char * __to ) const` [protected],[virtual]

Narrow char\_type array to char.

This virtual function converts each char\_type in the range [`__lo`,`__hi`) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `__dfault` is used instead.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__dfault</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

#### Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

**4.535.2.5** `template<typename _CharT> virtual const char_type* std::ctype< _CharT >::do_scan_is ( mask __m, const  
char_type * __lo, const char_type * __hi ) const` [protected],[virtual]

Find char\_type matching mask.

This function searches for and returns the first char\_type c in [`__lo`,`__hi`) for which is(`__m`,c) is true.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

#### Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

**Returns**

Pointer to a matching `char_type` if found, else `__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**4.535.2.6** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_scan_not ( mask __m, const char_type * __lo, const char_type * __hi ) const` `[protected]`, `[virtual]`

Find `char_type` not matching mask.

This function searches for and returns a pointer to the first `char_type` `c` of `[lo,hi)` for which `is(m,c)` is false.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

**Parameters**

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

**Returns**

Pointer to a non-matching `char_type` if found, else `__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**4.535.2.7** `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_tolower ( char_type __c ) const` `[protected]`, `[virtual]`

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

**Parameters**

<code>__c</code>	The <code>char_type</code> to convert.
------------------	--

**Returns**

The lowercase `char_type` if convertible, else `__c`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by `std::ctype<char>::tolower()`.

**4.535.2.8** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_tolower ( char_type * __lo, const char_type * __hi ) const` `[protected]`, `[virtual]`

Convert array to lowercase.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

#### Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

#### Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

**4.535.2.9** `template<typename _CharT > virtual char_type std::ctype< _CharT >::do_toupper ( char_type __c ) const`  
`[protected], [virtual]`

Convert to uppercase.

This virtual function converts the `char_type` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

#### Parameters

<code>__c</code>	The <code>char_type</code> to convert.
------------------	--

#### Returns

The uppercase `char_type` if convertible, else `__c`.

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

Referenced by `std::ctype< char >::toupper()`.

**4.535.2.10** `template<typename _CharT > virtual const char_type* std::ctype< _CharT >::do_toupper ( char_type * __lo, const char_type * __hi ) const`  
`[protected], [virtual]`

Convert array to uppercase.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to uppercase if possible. Other elements remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

#### Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

#### Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

**4.535.2.11** `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_widen ( char __c ) const`  
`[protected], [virtual]`

Widen char.

This virtual function converts the char to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

#### Returns

The converted char\_type

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by `std::ctype<char>::widen()`.

**4.535.2.12** `template<typename _CharT> virtual const char* std::ctype<_CharT>::do_widen ( const char * __lo, const char * __hi, char_type * __to ) const`  
`[protected], [virtual]`

Widen char array.

This function converts each char in the input to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

<code>__lo</code>	Pointer to start range.
<code>__hi</code>	Pointer to end of range.
<code>__to</code>	Pointer to the destination array.

#### Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**4.535.2.13** `template<typename _CharT> bool std::__ctype_abstract_base<_CharT>::is ( mask __m, char_type __c )`  
`const [inline], [inherited]`

Test char\_type classification.

This function finds a mask M for `__c` and compares it to mask `__m`. It does so by returning the value of `ctype<char_type>::do_is()`.

## Parameters

<code>__c</code>	The <code>char_type</code> to compare the mask of.
<code>__m</code>	The mask to compare against.

## Returns

`(M & __m) != 0.`

Definition at line 162 of file `locale_facets.h`.

**4.535.2.14** `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::is ( const char_type * __lo, const char_type * __hi, mask * __vec ) const` `[inline]`, `[inherited]`

Return a mask array.

This function finds the mask for each `char_type` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the char array. It does so by returning the value of `ctype<char_type>::do_is()`.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.

## Returns

`__hi.`

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

**4.535.2.15** `template<typename _CharT> char std::__ctype_abstract_base<_CharT>::narrow ( char_type __c, char __dfault ) const` `[inline]`, `[inherited]`

Narrow `char_type` to `char`.

This function converts the `char_type` to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead. It does so by returning `ctype<char_type>::do_narrow(__c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

<code>__c</code>	The <code>char_type</code> to convert.
<code>__dfault</code>	Char to return if conversion fails.

## Returns

The converted `char`.

Definition at line 324 of file `locale_facets.h`.

**4.535.2.16** `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::narrow ( const char_type * __lo, const char_type * __hi, char __dfault, char * __to ) const` `[inline]`, `[inherited]`

Narrow array to `char` array.

This function converts each `char_type` in the input to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `char_type` in the input that cannot be converted, `dfault` is used instead. It does so by returning `ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to)`.



Note: this is not what you want for codepage conversions. See `codecv` for that.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__dfault</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

## Returns

`__hi`.

Definition at line 346 of file locale\_facets.h.

**4.535.2.17** `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::scan_is ( mask __m, const char_type * __lo, const char_type * __hi ) const` `[inline]`, `[inherited]`

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char\_type>::do\_scan\_is().

## Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

## Returns

Pointer to matching char\_type if found, else `__hi`.

Definition at line 195 of file locale\_facets.h.

**4.535.2.18** `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::scan_not ( mask __m, const char_type * __lo, const char_type * __hi ) const` `[inline]`, `[inherited]`

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char\_type>::do\_scan\_not().

## Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to first char in range.
<code>__hi</code>	Pointer to end of range.

## Returns

Pointer to non-matching char if found, else `__hi`.

Definition at line 211 of file locale\_facets.h.

**4.535.2.19** `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::tolower ( char_type __c ) const` `[inline]`, `[inherited]`

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_tolower(c).

**Parameters**

<code>__c</code>	The <code>char_type</code> to convert.
------------------	--

**Returns**

The lowercase `char_type` if convertible, else `__c`.

Definition at line 254 of file `locale_facets.h`.

**4.535.2.20** `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::tolower ( char_type * __lo, const char_type * __hi ) const [inline],[inherited]`

Convert array to lowercase.

This function converts each `char_type` in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_tolower(__lo, __hi)`.

**Parameters**

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

**Returns**

`__hi`.

Definition at line 269 of file `locale_facets.h`.

**4.535.2.21** `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::toupper ( char_type __c ) const [inline],[inherited]`

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

**Parameters**

<code>__c</code>	The <code>char_type</code> to convert.
------------------	--

**Returns**

The uppercase `char_type` if convertible, else `__c`.

Definition at line 225 of file `locale_facets.h`.

**4.535.2.22** `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::toupper ( char_type * __lo, const char_type * __hi ) const [inline],[inherited]`

Convert array to uppercase.

This function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

## Returns

`__hi`.

Definition at line 240 of file locale\_facets.h.

**4.535.2.23** `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::widen ( char __c ) const`  
`[inline], [inherited]`

Widen char to char\_type.

This function converts the char argument to char\_type using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

## Returns

The converted char\_type.

Definition at line 286 of file locale\_facets.h.

**4.535.2.24** `template<typename _CharT> const char* std::__ctype_abstract_base<_CharT>::widen ( const char * __lo,`  
`const char * __hi, char_type * __to ) const` `[inline], [inherited]`

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__to</code>	Pointer to the destination array.

## Returns

`__hi`.

Definition at line 305 of file locale\_facets.h.

**4.535.3 Member Data Documentation**

**4.535.3.1** `template<typename _CharT> locale::id std::ctype<_CharT>::id` `[static]`

The facet id for `ctype<char_type>`

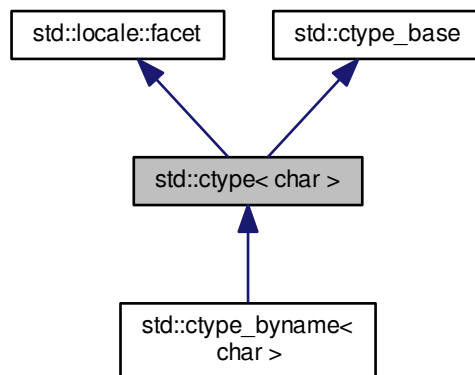
Definition at line 613 of file locale\_facets.h.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

#### 4.536 std::ctype< char > Class Template Reference

Inheritance diagram for std::ctype< char >:



##### Public Types

- typedef const int \* **\_\_to\_type**
- typedef char [char\\_type](#)
- typedef unsigned short **mask**

##### Public Member Functions

- [ctype](#) (const mask \* \_\_table=0, bool \_\_del=false, size\_t \_\_refs=0)
- [ctype](#) (\_\_c\_locale \_\_cloc, const mask \* \_\_table=0, bool \_\_del=false, size\_t \_\_refs=0)
- bool [is](#) (mask \_\_m, char \_\_c) const
- const char \* [is](#) (const char \* \_\_lo, const char \* \_\_hi, mask \* \_\_vec) const
- char [narrow](#) ([char\\_type](#) \_\_c, char \_\_default) const
- const [char\\_type](#) \* [narrow](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, char \_\_default, char \* \_\_to) const
- const char \* [scan\\_is](#) (mask \_\_m, const char \* \_\_lo, const char \* \_\_hi) const
- const char \* [scan\\_not](#) (mask \_\_m, const char \* \_\_lo, const char \* \_\_hi) const
- const mask \* [table](#) () const throw ()
- [char\\_type](#) [tolower](#) ([char\\_type](#) \_\_c) const
- const [char\\_type](#) \* [tolower](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- [char\\_type](#) [toupper](#) ([char\\_type](#) \_\_c) const
- const [char\\_type](#) \* [toupper](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- [char\\_type](#) [widen](#) (char \_\_c) const
- const char \* [widen](#) (const char \* \_\_lo, const char \* \_\_hi, [char\\_type](#) \* \_\_to) const

## Static Public Member Functions

- static const mask \* [classic\\_table](#) () throw ()

## Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const size\_t [table\\_size](#)
- static const mask **upper**
- static const mask **xdigit**

## Protected Member Functions

- virtual [~ctype](#) ()
- virtual char [do\\_narrow](#) (char\_type \_\_c, char \_\_dfault) const
- virtual const char\_type \* [do\\_narrow](#) (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_dfault, char \* \_\_to) const
- virtual char\_type [do\\_tolower](#) (char\_type \_\_c) const
- virtual const char\_type \* [do\\_tolower](#) (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type [do\\_toupper](#) (char\_type \_\_c) const
- virtual const char\_type \* [do\\_toupper](#) (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual [char\\_type do\\_widen](#) (char \_\_c) const
- virtual const char \* [do\\_widen](#) (const char \* \_\_lo, const char \* \_\_hi, char\_type \* \_\_to) const

## Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \* \_\_s)

## Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_ctype**
- bool **\_M\_del**
- char **\_M\_narrow** [1+static\_cast< unsigned char >(-1)]
- char **\_M\_narrow\_ok**
- const mask \* **\_M\_table**
- \_\_to\_type **\_M\_tolower**

- `__to_type_M_toupper`
- `char_M_widen` [1+static\_cast< unsigned char >(-1)]
- `char_M_widen_ok`

#### 4.536.1 Detailed Description

`template<>class std::ctype< char >`

The `ctype<char>` specialization.

This class defines classification and conversion functions for the `char` type. It gets used by `char` streams for many I/O operations. The `char` specialization provides a number of optimizations as well.

Definition at line 674 of file `locale_facets.h`.

#### 4.536.2 Member Typedef Documentation

##### 4.536.2.1 typedef `char std::ctype< char >::char_type`

Typedef for the template parameter `char`.

Definition at line 679 of file `locale_facets.h`.

#### 4.536.3 Constructor & Destructor Documentation

##### 4.536.3.1 `std::ctype< char >::ctype ( const mask * __table = 0, bool __del = false, size_t __refs = 0 )` [explicit]

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

<code>__table</code>	If non-zero, table is used as the per-char mask. Else <code>classic_table()</code> is used.
<code>__del</code>	If true, passes ownership of table to this facet.
<code>__refs</code>	Passed to the base facet class.

##### 4.536.3.2 `std::ctype< char >::ctype ( __c_locale __cloc, const mask * __table = 0, bool __del = false, size_t __refs = 0 )` [explicit]

Constructor performs static initialization.

This constructor is used to construct the initial C locale facet.

Parameters

<code>__cloc</code>	Handle to C locale data.
<code>__table</code>	If non-zero, table is used as the per-char mask.
<code>__del</code>	If true, passes ownership of table to this facet.
<code>__refs</code>	Passed to the base facet class.

##### 4.536.3.3 `virtual std::ctype< char >::~~ctype ( )` [protected],[virtual]

Destructor.

This function deletes `table()` if `del` was true in the constructor.

## 4.536.4 Member Function Documentation

## 4.536.4.1 static const mask\* std::ctype&lt; char &gt;::classic\_table ( ) throw [static]

Returns a pointer to the C locale mask table.

## 4.536.4.2 virtual char std::ctype&lt; char &gt;::do\_narrow ( char\_type \_\_c, char \_\_dfault ) const [inline], [protected], [virtual]

Narrow char.

This virtual function converts the char to char using the simplest reasonable transformation. If the conversion fails, *dfault* is returned instead. For an underived ctype<char> facet, *c* will be returned unchanged.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

<code>__c</code>	The char to convert.
<code>__dfault</code>	Char to return if conversion fails.

## Returns

The converted char.

Definition at line 1124 of file locale\_facets.h.

## 4.536.4.3 virtual const char\_type\* std::ctype&lt; char &gt;::do\_narrow ( const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_dfault, char \* \_\_to ) const [inline], [protected], [virtual]

Narrow char array to char array.

This virtual function converts each char in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived ctype<char> facet, the argument will be copied unchanged.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__dfault</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

## Returns

`__hi`.

Definition at line 1150 of file locale\_facets.h.

## 4.536.4.4 virtual char\_type std::ctype&lt; char &gt;::do\_tolower ( char\_type \_\_c ) const [protected], [virtual]

Convert to lowercase.



This virtual function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

#### Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

#### Returns

The lowercase char if convertible, else `__c`.

**4.536.4.5** `virtual const char_type* std::ctype< char >::do_tolower ( char_type * __lo, const char_type * __hi ) const`  
[protected], [virtual]

Convert array to lowercase.

This virtual function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

#### Parameters

<code>__lo</code>	Pointer to first char in range.
<code>__hi</code>	Pointer to end of range.

#### Returns

`__hi`.

**4.536.4.6** `virtual char_type std::ctype< char >::do_toupper ( char_type __c ) const` [protected], [virtual]

Convert to uppercase.

This virtual function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

#### Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

#### Returns

The uppercase char if convertible, else `__c`.

**4.536.4.7** `virtual const char_type* std::ctype< char >::do_toupper ( char_type * __lo, const char_type * __hi ) const`  
[protected], [virtual]

Convert array to uppercase.

This virtual function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

## Returns

`__hi`.

**4.536.4.8** `virtual char_type std::ctype< char >::do_widen ( char __c ) const` `[inline]`, `[protected]`, `[virtual]`

Widen char.

This virtual function converts the char to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be returned unchanged.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

## Returns

The converted character.

Definition at line 1075 of file locale\_facets.h.

**4.536.4.9** `virtual const char* std::ctype< char >::do_widen ( const char * __lo, const char * __hi, char_type * __to ) const` `[inline]`, `[protected]`, `[virtual]`

Widen char array.

This function converts each char in the range [lo,hi) to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be copied unchanged.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__to</code>	Pointer to the destination array.

## Returns

`__hi`.

Definition at line 1098 of file locale\_facets.h.

**4.536.4.10** `bool std::ctype< char >::is ( mask __m, char __c ) const` `[inline]`

Test char classification.

This function compares the mask table[c] to `__m`.

## Parameters

<code>__c</code>	The char to compare the mask of.
<code>__m</code>	The mask to compare against.

## Returns

True if `__m & table[__c]` is true, false otherwise.

Definition at line 43 of file `ctype_inline.h`.

**4.536.4.11** `const char * std::ctype< char >::is ( const char * __lo, const char * __hi, mask * __vec ) const` `[inline]`

Return a mask array.

This function finds the mask for each char in the range `[lo, hi)` and successively writes it to `vec`. `vec` must have as many elements as the char array.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.

## Returns

`__hi`.

Definition at line 48 of file `ctype_inline.h`.

**4.536.4.12** `char std::ctype< char >::narrow ( char_type __c, char __dfault ) const` `[inline]`

Narrow char.

This function converts the `char` to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead. For an undervied `ctype<char>` facet, `c` will be returned unchanged.

This function works as if it returns `ctype<char>::do_narrow(c)`. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

<code>__c</code>	The char to convert.
<code>__dfault</code>	Char to return if conversion fails.

## Returns

The converted character.

Definition at line 923 of file `locale_facets.h`.

References `std::ctype< _CharT >::do_narrow()`.

**4.536.4.13** `const char_type* std::ctype< char >::narrow ( const char_type * __lo, const char_type * __hi, char __dfault, char * __to ) const` `[inline]`

Narrow char array.

This function converts each char in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived ctype<char> facet, the argument will be copied unchanged.

This function works as if it returns ctype<char>::do\_narrow(lo, hi, dfault, to). do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__dfault</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

#### Returns

`__hi`.

Definition at line 956 of file locale\_facets.h.

References std::ctype<\_CharT>::do\_narrow().

4.536.4.14 `const char * std::ctype< char >::scan_is ( mask __m, const char * __lo, const char * __hi ) const` [inline]

Find char matching a mask.

This function searches for and returns the first char in [lo,hi) for which is(m,char) is true.

#### Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

#### Returns

Pointer to a matching char if found, else `__hi`.

Definition at line 57 of file ctype\_inline.h.

4.536.4.15 `const char * std::ctype< char >::scan_not ( mask __m, const char * __lo, const char * __hi ) const` [inline]

Find char not matching a mask.

This function searches for and returns a pointer to the first char in [\_\_lo,\_\_hi) for which is(m,char) is false.

#### Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

#### Returns

Pointer to a non-matching char if found, else `__hi`.

Definition at line 67 of file ctype\_inline.h.

**4.536.4.16** `const mask* std::ctype< char >::table ( ) const throw ( )` `[inline]`

Returns a pointer to the mask table provided to the constructor, or the default from `classic_table()` if none was provided.

Definition at line 974 of file `locale_facets.h`.

**4.536.4.17** `char_type std::ctype< char >::tolower ( char_type __c ) const` `[inline]`

Convert to lowercase.

This function converts the `char` argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`tolower()` acts as if it returns `ctype<char>::do_tolower(__c)`. `do_tolower()` must always return the same result for the same input.

#### Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

#### Returns

The lowercase char if convertible, else `__c`.

Definition at line 828 of file `locale_facets.h`.

References `std::ctype< _CharT >::do_tolower()`.

**4.536.4.18** `const char_type* std::ctype< char >::tolower ( char_type * __lo, const char_type * __hi ) const` `[inline]`

Convert array to lowercase.

This function converts each `char` in the range `[lo,hi)` to lowercase if possible. Other chars remain untouched.

`tolower()` acts as if it returns `ctype<char>::do_tolower(__lo, __hi)`. `do_tolower()` must always return the same result for the same input.

#### Parameters

<code>__lo</code>	Pointer to first char in range.
<code>__hi</code>	Pointer to end of range.

#### Returns

`__hi`.

Definition at line 845 of file `locale_facets.h`.

References `std::ctype< _CharT >::do_tolower()`.

**4.536.4.19** `char_type std::ctype< char >::toupper ( char_type __c ) const` `[inline]`

Convert to uppercase.

This function converts the `char` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`toupper()` acts as if it returns `ctype<char>::do_toupper(c)`. `do_toupper()` must always return the same result for the same input.

**Parameters**

<code>__c</code>	The char to convert.
------------------	----------------------

**Returns**

The uppercase char if convertible, else `__c`.

Definition at line 795 of file `locale_facets.h`.

References `std::ctype<_CharT>::do_toupper()`.

**4.536.4.20** `const char_type* std::ctype<char>::toupper ( char_type * __lo, const char_type * __hi ) const`  
`[inline]`

Convert array to uppercase.

This function converts each char in the range `[__lo,__hi)` to uppercase if possible. Other chars remain untouched.

`toupper()` acts as if it returns `ctype<char>::do_toupper(__lo, __hi)`. `do_toupper()` must always return the same result for the same input.

**Parameters**

<code>__lo</code>	Pointer to first char in range.
<code>__hi</code>	Pointer to end of range.

**Returns**

`__hi`.

Definition at line 812 of file `locale_facets.h`.

References `std::ctype<_CharT>::do_toupper()`.

**4.536.4.21** `char_type std::ctype<char>::widen ( char __c ) const` `[inline]`

Widen char.

This function converts the char to `char_type` using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be returned unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

<code>__c</code>	The char to convert.
------------------	----------------------

**Returns**

The converted character.

Definition at line 865 of file `locale_facets.h`.

References `std::ctype<_CharT>::do_widen()`.

**4.536.4.22** `const char* std::ctype<char>::widen ( const char * __lo, const char * __hi, char_type * __to ) const`  
`[inline]`

Widen char array.

This function converts each char in the input to char using the simplest reasonable transformation. For an undervived ctype<char> facet, the argument will be copied unchanged.

This function works as if it returns ctype<char>::do\_widen(c). do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

<code>__lo</code>	Pointer to first char in range.
<code>__hi</code>	Pointer to end of range.
<code>__to</code>	Pointer to the destination array.

#### Returns

`__hi`.

Definition at line 892 of file locale\_facets.h.

References std::ctype< \_CharT >::do\_widen().

### 4.536.5 Member Data Documentation

#### 4.536.5.1 locale::id std::ctype< char >::id [static]

The facet id for ctype<char>

Definition at line 696 of file locale\_facets.h.

#### 4.536.5.2 const size\_t std::ctype< char >::table\_size [static]

The size of the mask table. It is SCHAR\_MAX + 1.

Definition at line 698 of file locale\_facets.h.

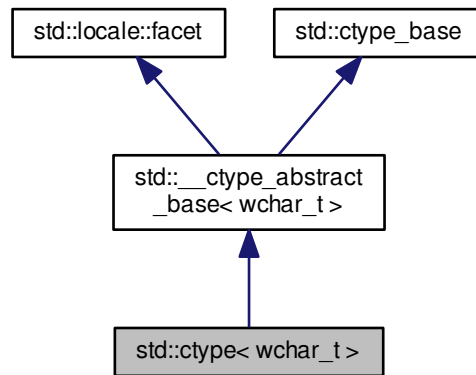
The documentation for this class was generated from the following files:

- [locale\\_facets.h](#)
- [ctype\\_inline.h](#)



#### 4.537 `std::ctype< wchar_t >` Class Template Reference

Inheritance diagram for `std::ctype< wchar_t >`:



##### Public Types

- `typedef const int * __to_type`
- `typedef wctype_t __wmask_type`
- `typedef wchar_t char_type`
- `typedef unsigned short mask`

##### Public Member Functions

- `ctype` (`size_t __refs=0`)
- `ctype` (`__c_locale __cloc, size_t __refs=0`)
- `bool is` (`mask __m, char_type __c`) `const`
- `const char_type * is` (`const char_type * __lo, const char_type * __hi, mask * __vec`) `const`
- `char narrow` (`char_type __c, char __dfault`) `const`
- `const char_type * narrow` (`const char_type * __lo, const char_type * __hi, char __dfault, char * __to`) `const`
- `const char_type * scan_is` (`mask __m, const char_type * __lo, const char_type * __hi`) `const`
- `const char_type * scan_not` (`mask __m, const char_type * __lo, const char_type * __hi`) `const`
- `char_type tolower` (`char_type __c`) `const`
- `const char_type * tolower` (`char_type * __lo, const char_type * __hi`) `const`
- `char_type toupper` (`char_type __c`) `const`
- `const char_type * toupper` (`char_type * __lo, const char_type * __hi`) `const`
- `char_type widen` (`char __c`) `const`
- `const char * widen` (`const char * __lo, const char * __hi, char_type * __to`) `const`

## Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

## Protected Member Functions

- virtual [~ctype](#) ()
- [\\_\\_wmask\\_type](#) **\_M\_convert\_to\_wmask** (const mask \_\_m) const throw ()
- void **\_M\_initialize\_ctype** () throw ()
- virtual bool [do\\_is](#) (mask \_\_m, [char\\_type](#) \_\_c) const
- virtual const [char\\_type](#) \* [do\\_is](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, mask \* \_\_vec) const
- virtual [char](#) [do\\_narrow](#) ([char\\_type](#) \_\_c, [char](#) \_\_default) const
- virtual const [char\\_type](#) \* [do\\_narrow](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, [char](#) \_\_default, [char](#) \* \_\_to) const
- virtual const [char\\_type](#) \* [do\\_scan\\_is](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual const [char\\_type](#) \* [do\\_scan\\_not](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual [char\\_type](#) [do\\_tolower](#) ([char\\_type](#) \_\_c) const
- virtual const [char\\_type](#) \* [do\\_tolower](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual [char\\_type](#) [do\\_toupper](#) ([char\\_type](#) \_\_c) const
- virtual const [char\\_type](#) \* [do\\_toupper](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual [char\\_type](#) [do\\_widen](#) ([char](#) \_\_c) const
- virtual const [char](#) \* [do\\_widen](#) (const [char](#) \* \_\_lo, const [char](#) \* \_\_hi, [char\\_type](#) \* \_\_to) const

## Static Protected Member Functions

- static [\\_\\_c\\_locale](#) **\_S\_clone\_c\_locale** ([\\_\\_c\\_locale](#) & \_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** ([\\_\\_c\\_locale](#) & \_\_cloc, const [char](#) \* \_\_s, [\\_\\_c\\_locale](#) \_\_old=0)
- static void **\_S\_destroy\_c\_locale** ([\\_\\_c\\_locale](#) & \_\_cloc)
- static [\\_\\_c\\_locale](#) **\_S\_get\_c\_locale** ()
- static const [char](#) \* **\_S\_get\_c\_name** () throw ()
- static [\\_\\_c\\_locale](#) **\_S\_lc\_ctype\_c\_locale** ([\\_\\_c\\_locale](#) \_\_cloc, const [char](#) \* \_\_s)

## Protected Attributes

- mask **\_M\_bit** [16]
- [\\_\\_c\\_locale](#) **\_M\_c\_locale\_ctype**
- [char](#) **\_M\_narrow** [128]
- bool **\_M\_narrow\_ok**
- [wint\\_t](#) **\_M\_widen** [1+static\_cast< unsigned char >(-1)]
- [\\_\\_wmask\\_type](#) **\_M\_wmask** [16]

#### 4.537.1 Detailed Description

`template<>class std::ctype< wchar_t >`

The `ctype<wchar_t>` specialization.

This class defines classification and conversion functions for the `wchar_t` type. It gets used by `wchar_t` streams for many I/O operations. The `wchar_t` specialization provides a number of optimizations as well.

`ctype<wchar_t>` inherits its public methods from `__ctype_abstract_base<wchar_t>`.

Definition at line 1175 of file `locale_facets.h`.

#### 4.537.2 Member Typedef Documentation

4.537.2.1 `typedef wchar_t std::ctype< wchar_t >::char_type`

Typedef for the template parameter `wchar_t`.

Definition at line 1180 of file `locale_facets.h`.

#### 4.537.3 Constructor & Destructor Documentation

4.537.3.1 `std::ctype< wchar_t >::ctype ( size_t __refs = 0 ) [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

4.537.3.2 `std::ctype< wchar_t >::ctype ( __c_locale __cloc, size_t __refs = 0 ) [explicit]`

Constructor performs static initialization.

This constructor is used to construct the initial C locale facet.

Parameters

<code>__cloc</code>	Handle to C locale data.
<code>__refs</code>	Passed to the base facet class.

4.537.3.3 `virtual std::ctype< wchar_t >::~~ctype ( ) [protected],[virtual]`

Destructor.

#### 4.537.4 Member Function Documentation

4.537.4.1 `virtual bool std::ctype< wchar_t >::do_is ( mask __m, char_type __c ) const [protected],[virtual]`

Test `wchar_t` classification.

This function finds a mask `M` for `c` and compares it to mask `m`.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

## Parameters

<code>__c</code>	The <code>wchar_t</code> to find the mask of.
<code>__m</code>	The mask to compare against.

## Returns

`(M & __m) != 0.`

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**4.537.4.2** `virtual const char_type* std::ctype< wchar_t >::do_is ( const char_type * __lo, const char_type * __hi, mask * __vec ) const` [protected],[virtual]

Return a mask array.

This function finds the mask for each `wchar_t` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the input.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.

## Returns

`__hi.`

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**4.537.4.3** `virtual char std::ctype< wchar_t >::do_narrow ( char_type __c, char __dfault ) const` [protected],[virtual]

Narrow `wchar_t` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead. For an underived `ctype<wchar_t>` facet, `c` will be cast to `char` and returned.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

<code>__c</code>	The <code>wchar_t</code> to convert.
<code>__dfault</code>	Char to return if conversion fails.

## Returns

The converted `char`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

4.537.4.4 **virtual const char\_type\* std::ctype< wchar\_t >::do\_narrow ( const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_default, char \* \_\_to ) const** [protected], [virtual]

Narrow wchar\_t array to char array.

This virtual function converts each wchar\_t in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any wchar\_t in the input that cannot be converted, *default* is used instead. For an underived ctype<wchar\_t> facet, the argument will be copied, casting each element to char.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__default</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

#### Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

4.537.4.5 **virtual const char\_type\* std::ctype< wchar\_t >::do\_scan\_is ( mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi ) const** [protected], [virtual]

Find wchar\_t matching mask.

This function searches for and returns the first wchar\_t c in [\_\_lo,\_\_hi) for which is(\_\_m,c) is true.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

#### Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

#### Returns

Pointer to a matching wchar\_t if found, else `__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

4.537.4.6 **virtual const char\_type\* std::ctype< wchar\_t >::do\_scan\_not ( mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi ) const** [protected], [virtual]

Find wchar\_t not matching mask.

This function searches for and returns a pointer to the first wchar\_t c of [\_\_lo,\_\_hi) for which is(\_\_m,c) is false.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

## Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

## Returns

Pointer to a non-matching `wchar_t` if found, else `__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**4.537.4.7** `virtual char_type std::ctype< wchar_t >::do_tolower ( char_type __c ) const` `[protected]`, `[virtual]`

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

## Parameters

<code>__c</code>	The <code>wchar_t</code> to convert.
------------------	--------------------------------------

## Returns

The lowercase `wchar_t` if convertible, else `__c`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**4.537.4.8** `virtual const char_type* std::ctype< wchar_t >::do_tolower ( char_type * __lo, const char_type * __hi ) const` `[protected]`, `[virtual]`

Convert array to lowercase.

This virtual function converts each `wchar_t` in the range `[lo,hi)` to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

## Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**4.537.4.9** `virtual char_type std::ctype< wchar_t >::do_toupper ( char_type __c ) const` `[protected]`, `[virtual]`

Convert to uppercase.

This virtual function converts the `wchar_t` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.



## Parameters

<code>__c</code>	The <code>wchar_t</code> to convert.
------------------	--------------------------------------

## Returns

The uppercase `wchar_t` if convertible, else `__c`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**4.537.4.10** `virtual const char_type* std::ctype< wchar_t >::do_toupper ( char_type * __lo, const char_type * __hi ) const` `[protected]`, `[virtual]`

Convert array to uppercase.

This virtual function converts each `wchar_t` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

## Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**4.537.4.11** `virtual char_type std::ctype< wchar_t >::do_widen ( char __c ) const` `[protected]`, `[virtual]`

Widen char to `wchar_t`.

This virtual function converts the char to `wchar_t` using the simplest reasonable transformation. For an underived `ctype<wchar_t>` facet, the argument will be cast to `wchar_t`.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

## Returns

The converted `wchar_t`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**4.537.4.12** `virtual const char* std::ctype< wchar_t >::do_widen ( const char * __lo, const char * __hi, char_type * __to ) const` `[protected]`, `[virtual]`

Widen char array to `wchar_t` array.

This function converts each char in the input to `wchar_t` using the simplest reasonable transformation. For an underived `ctype<wchar_t>` facet, the argument will be copied, casting each element to `wchar_t`.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

<code>__lo</code>	Pointer to start range.
<code>__hi</code>	Pointer to end of range.
<code>__to</code>	Pointer to the destination array.

#### Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**4.537.4.13** `bool std::__ctype_abstract_base< wchar_t >::is ( mask __m, char_type __c ) const` `[inline]`,  
`[inherited]`

Test char\_type classification.

This function finds a mask M for `__c` and compares it to mask `__m`. It does so by returning the value of `ctype<char_type>::do_is()`.

#### Parameters

<code>__c</code>	The char_type to compare the mask of.
<code>__m</code>	The mask to compare against.

#### Returns

`(M & __m) != 0`.

Definition at line 162 of file locale\_facets.h.

References `std::__ctype_abstract_base< _CharT >::do_is()`.

**4.537.4.14** `const char_type* std::__ctype_abstract_base< wchar_t >::is ( const char_type * __lo, const char_type * __hi, mask * __vec ) const` `[inline]`,`[inherited]`

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of `ctype<char_type>::do_is()`.

#### Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.

#### Returns

`__hi`.

Definition at line 179 of file locale\_facets.h.

References `std::__ctype_abstract_base< _CharT >::do_is()`.

**4.537.4.15** `char std::__ctype_abstract_base<wchar_t>::narrow ( char_type __c, char __default ) const` `[inline], [inherited]`

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, default is returned instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_c).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

<code>__c</code>	The char_type to convert.
<code>__default</code>	Char to return if conversion fails.

#### Returns

The converted char.

Definition at line 324 of file locale\_facets.h.

References std::\_\_ctype\_abstract\_base<\_CharT>::do\_narrow().

**4.537.4.16** `const char_type* std::__ctype_abstract_base<wchar_t>::narrow ( const char_type * __lo, const char_type * __hi, char __default, char * __to ) const` `[inline], [inherited]`

Narrow array to char array.

This function converts each char\_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char\_type in the input that cannot be converted, default is used instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_lo, \_\_hi, \_\_default, \_\_to).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__default</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

#### Returns

`__hi`.

Definition at line 346 of file locale\_facets.h.

References std::\_\_ctype\_abstract\_base<\_CharT>::do\_narrow().

**4.537.4.17** `const char_type* std::__ctype_abstract_base<wchar_t>::scan_is ( mask __m, const char_type * __lo, const char_type * __hi ) const` `[inline], [inherited]`

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char\_type>::do\_scan\_is().

## Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

## Returns

Pointer to matching char\_type if found, else `__hi`.

Definition at line 195 of file locale\_facets.h.

References std::ctype\_abstract\_base< \_CharT >::do\_scan\_is().

**4.537.4.18** `const char_type* std::ctype_abstract_base< wchar_t >::scan_not ( mask __m, const char_type * __lo, const char_type * __hi ) const` [inline],[inherited]

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char\_type>::do\_scan\_not().

## Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to first char in range.
<code>__hi</code>	Pointer to end of range.

## Returns

Pointer to non-matching char if found, else `__hi`.

Definition at line 211 of file locale\_facets.h.

References std::ctype\_abstract\_base< \_CharT >::do\_scan\_not().

**4.537.4.19** `char_type std::ctype_abstract_base< wchar_t >::tolower ( char_type __c ) const` [inline],[inherited]

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_tolower(c).

## Parameters

<code>__c</code>	The char_type to convert.
------------------	---------------------------

## Returns

The lowercase char\_type if convertible, else `__c`.

Definition at line 254 of file locale\_facets.h.

References std::ctype\_abstract\_base< \_CharT >::do\_tolower().

**4.537.4.20** `const char_type* std::ctype_abstract_base< wchar_t >::tolower ( char_type * __lo, const char_type * __hi ) const` [inline],[inherited]

Convert array to lowercase.

This function converts each `char_type` in the range `[__lo, __hi)` to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_tolower(__lo, __hi)`.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

## Returns

`__hi`.

Definition at line 269 of file locale\_facets.h.

References `std::__ctype_abstract_base< _CharT >::do_tolower()`.

**4.537.4.21** `char_type std::__ctype_abstract_base< wchar_t >::toupper ( char_type __c ) const` `[inline]`, `[inherited]`

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

## Parameters

<code>__c</code>	The <code>char_type</code> to convert.
------------------	--

## Returns

The uppercase `char_type` if convertible, else `__c`.

Definition at line 225 of file locale\_facets.h.

References `std::__ctype_abstract_base< _CharT >::do_toupper()`.

**4.537.4.22** `const char_type* std::__ctype_abstract_base< wchar_t >::toupper ( char_type * __lo, const char_type * __hi ) const` `[inline]`, `[inherited]`

Convert array to uppercase.

This function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

## Returns

`__hi`.

Definition at line 240 of file locale\_facets.h.

References `std::__ctype_abstract_base< _CharT >::do_toupper()`.

**4.537.4.23** `char_type std::__ctype_abstract_base< wchar_t >::widen ( char __c ) const` `[inline]`, `[inherited]`

Widen char to `char_type`.

This function converts the `char` argument to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

<code>__c</code>	The char to convert.
------------------	----------------------

**Returns**

The converted `char_type`.

Definition at line 286 of file `locale_facets.h`.

References `std::__ctype_abstract_base<_CharT>::do_widen()`.

**4.537.4.24** `const char* std::__ctype_abstract_base<wchar_t>::widen ( const char * __lo, const char * __hi, char_type * __to ) const` `[inline],[inherited]`

Widen array to `char_type`.

This function converts each char in the input to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__to</code>	Pointer to the destination array.

**Returns**

`__hi`.

Definition at line 305 of file `locale_facets.h`.

References `std::__ctype_abstract_base<_CharT>::do_widen()`.

**4.537.5 Member Data Documentation**

**4.537.5.1** `locale::id std::ctype<wchar_t>::id` `[static]`

The facet id for `ctype<wchar_t>`

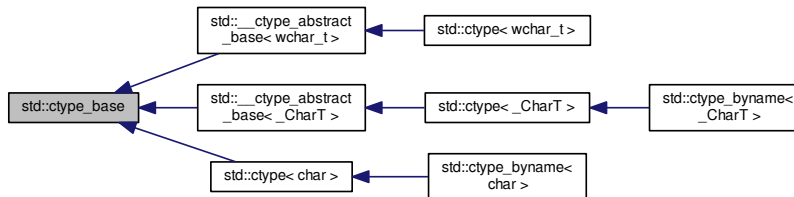
Definition at line 1198 of file `locale_facets.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 4.538 std::ctype\_base Struct Reference

Inheritance diagram for std::ctype\_base:



## Public Types

- typedef const int \* **\_\_to\_type**
- typedef unsigned short **mask**

## Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

## 4.538.1 Detailed Description

Base class for ctype.

Definition at line 41 of file ctype\_base.h.

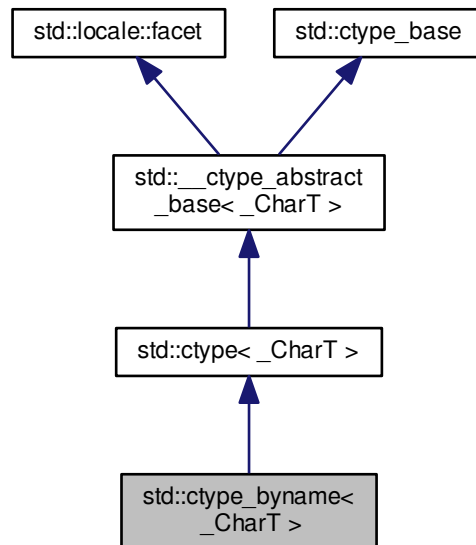
The documentation for this struct was generated from the following file:

- [ctype\\_base.h](#)



#### 4.539 `std::ctype_bname<_CharT>` Class Template Reference

Inheritance diagram for `std::ctype_bname<_CharT>`:



#### Public Types

- `typedef const int * __to_type`
- `typedef _CharT char_type`
- `typedef ctype<\_CharT>::mask mask`

#### Public Member Functions

- **`ctype_bname`** (`const char * __s, size_t __refs=0`)
- `bool is (mask __m, char\_type __c) const`
- `const char\_type * is (const char\_type * __lo, const char\_type * __hi, mask * __vec) const`
- `char narrow (char\_type __c, char __dfault) const`
- `const char\_type * narrow (const char\_type * __lo, const char\_type * __hi, char __dfault, char * __to) const`
- `const char\_type * scan_is (mask __m, const char\_type * __lo, const char\_type * __hi) const`
- `const char\_type * scan_not (mask __m, const char\_type * __lo, const char\_type * __hi) const`
- `char\_type tolower (char\_type __c) const`
- `const char\_type * tolower (char\_type * __lo, const char\_type * __hi) const`
- `char\_type toupper (char\_type __c) const`
- `const char\_type * toupper (char\_type * __lo, const char\_type * __hi) const`
- `char\_type widen (char __c) const`
- `const char * widen (const char * __lo, const char * __hi, char\_type * __to) const`

## Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

## Protected Member Functions

- virtual bool [do\\_is](#) (mask \_\_m, [char\\_type](#) \_\_c) const
- virtual const [char\\_type](#) \* [do\\_is](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, mask \* \_\_vec) const
- virtual [char](#) [do\\_narrow](#) ([char\\_type](#), [char](#) \_\_default) const
- virtual const [char\\_type](#) \* [do\\_narrow](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, [char](#) \_\_default, [char](#) \* \_\_to) const
- virtual const [char\\_type](#) \* [do\\_scan\\_is](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual const [char\\_type](#) \* [do\\_scan\\_not](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual [char\\_type](#) [do\\_tolower](#) ([char\\_type](#) \_\_c) const
- virtual const [char\\_type](#) \* [do\\_tolower](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual [char\\_type](#) [do\\_toupper](#) ([char\\_type](#) \_\_c) const
- virtual const [char\\_type](#) \* [do\\_toupper](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual [char\\_type](#) [do\\_widen](#) ([char](#) \_\_c) const
- virtual const [char](#) \* [do\\_widen](#) (const [char](#) \* \_\_lo, const [char](#) \* \_\_hi, [char\\_type](#) \* \_\_dest) const

## Static Protected Member Functions

- static [\\_\\_c\\_locale](#) [\\_S\\_clone\\_c\\_locale](#) ([\\_\\_c\\_locale](#) & \_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) ([\\_\\_c\\_locale](#) & \_\_cloc, const [char](#) \* \_\_s, [\\_\\_c\\_locale](#) \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) ([\\_\\_c\\_locale](#) & \_\_cloc)
- static [\\_\\_c\\_locale](#) [\\_S\\_get\\_c\\_locale](#) ()
- static const [char](#) \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static [\\_\\_c\\_locale](#) [\\_S\\_lc\\_ctype\\_c\\_locale](#) ([\\_\\_c\\_locale](#) \_\_cloc, const [char](#) \* \_\_s)

## 4.539.1 Detailed Description

```
template<typename _CharT>class std::ctype_byname<_CharT>
```

class [ctype\\_byname](#) [22.2.1.2].

Definition at line 1467 of file [locale\\_facets.h](#).

#### 4.539.2 Member Function Documentation

4.539.2.1 `template<typename _CharT> virtual bool std::ctype<_CharT>::do_is ( mask __m, char_type __c ) const`  
`[protected], [virtual], [inherited]`

Test `char_type` classification.

This function finds a mask `M` for `c` and compares it to mask `m`.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

##### Parameters

<code>__c</code>	The <code>char_type</code> to find the mask of.
<code>__m</code>	The mask to compare against.

##### Returns

`(M & __m) != 0.`

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

4.539.2.2 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_is ( const char_type * __lo, const char_type * __hi, mask * __vec ) const`  
`[protected], [virtual], [inherited]`

Return a mask array.

This function finds the mask for each `char_type` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the input.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

##### Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.

##### Returns

`__hi.`

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

4.539.2.3 `template<typename _CharT> virtual char std::ctype<_CharT>::do_narrow ( char_type __c, char __dfault ) const`  
`[protected], [virtual], [inherited]`

Narrow `char_type` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

<code>__c</code>	The char_type to convert.
<code>__default</code>	Char to return if conversion fails.

## Returns

The converted char.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by `std::ctype<char>::narrow()`.

**4.539.2.4** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_narrow ( const char_type * __lo, const char_type * __hi, char __default, char * __to ) const` [protected], [virtual], [inherited]

Narrow char\_type array to char.

This virtual function converts each char\_type in the range [`__lo`,`__hi`) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `__default` is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__default</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

## Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**4.539.2.5** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_scan_is ( mask __m, const char_type * __lo, const char_type * __hi ) const` [protected], [virtual], [inherited]

Find char\_type matching mask.

This function searches for and returns the first char\_type `c` in [`__lo`,`__hi`) for which `is(__m,c)` is true.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

## Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.

<code>__hi</code>	Pointer to end of range.
-------------------	--------------------------

**Returns**

Pointer to a matching `char_type` if found, else `__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**4.539.2.6** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_scan_not ( mask __m, const char_type * __lo, const char_type * __hi ) const` `[protected]`, `[virtual]`, `[inherited]`

Find `char_type` not matching mask.

This function searches for and returns a pointer to the first `char_type` `c` of `[lo,hi)` for which `is(m,c)` is false.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

**Parameters**

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

**Returns**

Pointer to a non-matching `char_type` if found, else `__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**4.539.2.7** `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_tolower ( char_type __c ) const` `[protected]`, `[virtual]`, `[inherited]`

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

**Parameters**

<code>__c</code>	The <code>char_type</code> to convert.
------------------	--

**Returns**

The lowercase `char_type` if convertible, else `__c`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by `std::ctype<char>::tolower()`.

**4.539.2.8** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_tolower ( char_type * __lo, const char_type * __hi ) const` `[protected]`, `[virtual]`, `[inherited]`

Convert array to lowercase.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

## Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**4.539.2.9** `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_toupper ( char_type __c ) const`  
`[protected], [virtual], [inherited]`

Convert to uppercase.

This virtual function converts the `char_type` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

## Parameters

<code>__c</code>	The <code>char_type</code> to convert.
------------------	--

## Returns

The uppercase `char_type` if convertible, else `__c`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by `std::ctype<char>::toupper()`.

**4.539.2.10** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_toupper ( char_type * __lo,`  
`const char_type * __hi ) const` `[protected], [virtual], [inherited]`

Convert array to uppercase.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to uppercase if possible. Other elements remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

## Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**4.539.2.11** `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_widen ( char __c ) const`  
`[protected], [virtual], [inherited]`

Widen char.

This virtual function converts the char to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

#### Returns

The converted char\_type

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by `std::ctype<char>::widen()`.

**4.539.2.12** `template<typename _CharT> virtual const char* std::ctype<_CharT>::do_widen ( const char * __lo, const char * __hi, char_type * __to ) const` `[protected]`, `[virtual]`, `[inherited]`

Widen char array.

This function converts each char in the input to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

<code>__lo</code>	Pointer to start range.
<code>__hi</code>	Pointer to end of range.
<code>__to</code>	Pointer to the destination array.

#### Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**4.539.2.13** `template<typename _CharT> bool std::__ctype_abstract_base<_CharT>::is ( mask __m, char_type __c ) const` `[inline]`, `[inherited]`

Test char\_type classification.

This function finds a mask M for `__c` and compares it to mask `__m`. It does so by returning the value of `ctype<char_type>::do_is()`.

#### Parameters

<code>__c</code>	The char_type to compare the mask of.
<code>__m</code>	The mask to compare against.

#### Returns

`(M & __m) != 0`.

Definition at line 162 of file locale\_facets.h.



4.539.2.14 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::is ( const char_type  
* __lo, const char_type * __hi, mask * __vec ) const [inline],[inherited]`

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char\_type>::do\_is().

#### Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.

#### Returns

`__hi`.

Definition at line 179 of file locale\_facets.h.

4.539.2.15 `template<typename _CharT> char std::__ctype_abstract_base<_CharT>::narrow ( char_type __c, char  
__default ) const [inline],[inherited]`

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, default is returned instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_c).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

<code>__c</code>	The char_type to convert.
<code>__default</code>	Char to return if conversion fails.

#### Returns

The converted char.

Definition at line 324 of file locale\_facets.h.

4.539.2.16 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::narrow ( const  
char_type * __lo, const char_type * __hi, char __default, char * __to ) const [inline],[inherited]`

Narrow array to char array.

This function converts each char\_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char\_type in the input that cannot be converted, default is used instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_lo, \_\_hi, \_\_default, \_\_to).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

<code>__lo</code>	Pointer to start of range.
-------------------	----------------------------

<code>__hi</code>	Pointer to end of range.
<code>__dfault</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

**Returns**`__hi`.

Definition at line 346 of file locale\_facets.h.

4.539.2.17 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::scan_is ( mask __m, const char_type * __lo, const char_type * __hi ) const` `[inline]`,`[inherited]`

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char\_type>::do\_scan\_is().

**Parameters**

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

**Returns**Pointer to matching char\_type if found, else `__hi`.

Definition at line 195 of file locale\_facets.h.

4.539.2.18 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::scan_not ( mask __m, const char_type * __lo, const char_type * __hi ) const` `[inline]`,`[inherited]`

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char\_type>::do\_scan\_not().

**Parameters**

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to first char in range.
<code>__hi</code>	Pointer to end of range.

**Returns**Pointer to non-matching char if found, else `__hi`.

Definition at line 211 of file locale\_facets.h.

4.539.2.19 `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::tolower ( char_type __c ) const` `[inline]`,`[inherited]`

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_tolower(c).

**Parameters**

<code>__c</code>	The <code>char_type</code> to convert.
------------------	--

**Returns**

The lowercase `char_type` if convertible, else `__c`.

Definition at line 254 of file `locale_facets.h`.

```
4.539.2.20  template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::tolower ( char_type
            * __lo, const char_type * __hi ) const    [inline],[inherited]
```

Convert array to lowercase.

This function converts each `char_type` in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_tolower(__lo, __hi)`.

**Parameters**

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

**Returns**

`__hi`.

Definition at line 269 of file `locale_facets.h`.

```
4.539.2.21  template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::toupper ( char_type __c )
            const    [inline],[inherited]
```

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

**Parameters**

<code>__c</code>	The <code>char_type</code> to convert.
------------------	--

**Returns**

The uppercase `char_type` if convertible, else `__c`.

Definition at line 225 of file `locale_facets.h`.

```
4.539.2.22  template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::toupper ( char_type
            * __lo, const char_type * __hi ) const    [inline],[inherited]
```

Convert array to uppercase.

This function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

## Returns

`__hi`.

Definition at line 240 of file locale\_facets.h.

**4.539.2.23** `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::widen ( char __c ) const`  
`[inline], [inherited]`

Widen char to char\_type.

This function converts the char argument to char\_type using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

## Returns

The converted char\_type.

Definition at line 286 of file locale\_facets.h.

**4.539.2.24** `template<typename _CharT> const char* std::__ctype_abstract_base<_CharT>::widen ( const char * __lo,`  
`const char * __hi, char_type * __to ) const` `[inline], [inherited]`

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__to</code>	Pointer to the destination array.

## Returns

`__hi`.

Definition at line 305 of file locale\_facets.h.

## 4.539.3 Member Data Documentation

**4.539.3.1** `template<typename _CharT> locale::id std::ctype<_CharT>::id` `[static], [inherited]`

The facet id for `ctype<char_type>`

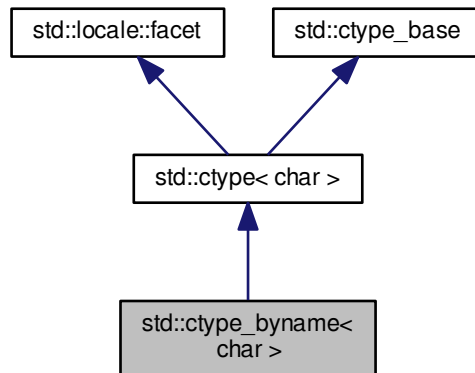
Definition at line 613 of file locale\_facets.h.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

#### 4.540 std::ctype\_byname< char > Class Template Reference

Inheritance diagram for std::ctype\_byname< char >:



##### Public Types

- typedef const int \* **\_\_to\_type**
- typedef char [char\\_type](#)
- typedef unsigned short **mask**

##### Public Member Functions

- **ctype\_byname** (const char \* \_\_s, size\_t \_\_refs=0)
- bool **is** (mask \_\_m, char \_\_c) const
- const char \* **is** (const char \* \_\_lo, const char \* \_\_hi, mask \* \_\_vec) const
- char **narrow** (char\_type \_\_c, char \_\_default) const
- const char\_type \* **narrow** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_default, char \* \_\_to) const
- const char \* **scan\_is** (mask \_\_m, const char \* \_\_lo, const char \* \_\_hi) const
- const char \* **scan\_not** (mask \_\_m, const char \* \_\_lo, const char \* \_\_hi) const
- const mask \* **table** () const throw ()
- char\_type **tolower** (char\_type \_\_c) const
- const char\_type \* **tolower** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **toupper** (char\_type \_\_c) const
- const char\_type \* **toupper** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **widen** (char \_\_c) const
- const char \* **widen** (const char \* \_\_lo, const char \* \_\_hi, char\_type \* \_\_to) const

## Static Public Member Functions

- static const mask \* [classic\\_table](#) () throw ()

## Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const size\_t [table\\_size](#)
- static const mask **upper**
- static const mask **xdigit**

## Protected Member Functions

- virtual char [do\\_narrow](#) (char\_type \_\_c, char \_\_dfault) const
- virtual const char\_type \* [do\\_narrow](#) (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_dfault, char \* \_\_to) const
- virtual char\_type [do\\_tolower](#) (char\_type \_\_c) const
- virtual const char\_type \* [do\\_tolower](#) (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type [do\\_toupper](#) (char\_type \_\_c) const
- virtual const char\_type \* [do\\_toupper](#) (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type [do\\_widen](#) (char \_\_c) const
- virtual const char \* [do\\_widen](#) (const char \* \_\_lo, const char \* \_\_hi, char\_type \* \_\_to) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale & \_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale & \_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale & \_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \* \_\_s)

## Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_ctype**
- bool **\_M\_del**
- char **\_M\_narrow** [1+static\_cast< unsigned char >(-1)]
- char **\_M\_narrow\_ok**
- const mask \* **\_M\_table**
- \_\_to\_type **\_M\_tolower**
- \_\_to\_type **\_M\_toupper**
- char **\_M\_widen** [1+static\_cast< unsigned char >(-1)]
- char **\_M\_widen\_ok**

#### 4.540.1 Detailed Description

`template<>class std::ctype_byname< char >`

22.2.1.4 Class `ctype_byname` specializations.

Definition at line 1482 of file `locale_facets.h`.

#### 4.540.2 Member Typedef Documentation

4.540.2.1 `typedef char std::ctype< char >::char_type` `[inherited]`

Typedef for the template parameter `char`.

Definition at line 679 of file `locale_facets.h`.

#### 4.540.3 Member Function Documentation

4.540.3.1 `static const mask* std::ctype< char >::classic_table ( ) throw` `[static],[inherited]`

Returns a pointer to the C locale mask table.

4.540.3.2 `virtual char std::ctype< char >::do_narrow ( char_type __c, char __dfault ) const` `[inline],[protected],[virtual],[inherited]`

Narrow `char`.

This virtual function converts the `char` to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead. For an underived `ctype<char>` facet, `c` will be returned unchanged.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

##### Parameters

<code>__c</code>	The <code>char</code> to convert.
<code>__dfault</code>	Char to return if conversion fails.

##### Returns

The converted `char`.

Definition at line 1124 of file `locale_facets.h`.

4.540.3.3 `virtual const char_type* std::ctype< char >::do_narrow ( const char_type * __lo, const char_type * __hi, char __dfault, char * __to ) const` `[inline],[protected],[virtual],[inherited]`

Narrow `char` array to `char` array.

This virtual function converts each `char` in the range `[lo,hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `char` in the input that cannot be converted, `dfault` is used instead. For an underived `ctype<char>` facet, the argument will be copied unchanged.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__dfault</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

## Returns

`__hi`.

Definition at line 1150 of file locale\_facets.h.

4.540.3.4 **virtual char\_type std::ctype< char >::do\_tolower ( char\_type \_\_c ) const** [protected],[virtual],[inherited]

Convert to lowercase.

This virtual function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

## Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

## Returns

The lowercase char if convertible, else `__c`.

4.540.3.5 **virtual const char\_type\* std::ctype< char >::do\_tolower ( char\_type \* \_\_lo, const char\_type \* \_\_hi ) const** [protected],[virtual],[inherited]

Convert array to lowercase.

This virtual function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

## Parameters

<code>__lo</code>	Pointer to first char in range.
<code>__hi</code>	Pointer to end of range.

## Returns

`__hi`.

4.540.3.6 **virtual char\_type std::ctype< char >::do\_toupper ( char\_type \_\_c ) const** [protected],[virtual],[inherited]

Convert to uppercase.

This virtual function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.



`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

## Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

## Returns

The uppercase char if convertible, else `__c`.

**4.540.3.7** `virtual const char_type* std::ctype< char >::do_toupper ( char_type * __lo, const char_type * __hi ) const` `[protected]`, `[virtual]`, `[inherited]`

Convert array to uppercase.

This virtual function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

## Returns

`__hi`.

**4.540.3.8** `virtual char_type std::ctype< char >::do_widen ( char __c ) const` `[inline]`, `[protected]`, `[virtual]`, `[inherited]`

Widen char.

This virtual function converts the char to char using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be returned unchanged.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

## Returns

The converted character.

Definition at line 1075 of file `locale_facets.h`.

**4.540.3.9** `virtual const char* std::ctype< char >::do_widen ( const char * __lo, const char * __hi, char_type * __to ) const` `[inline]`, `[protected]`, `[virtual]`, `[inherited]`

Widen char array.

This function converts each char in the range [lo,hi) to char using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be copied unchanged.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__to</code>	Pointer to the destination array.

**Returns**

`__hi`.

Definition at line 1098 of file `locale_facets.h`.

**4.540.3.10** `bool std::ctype< char >::is ( mask __m, char __c ) const` `[inline],[inherited]`

Test char classification.

This function compares the mask table[c] to `__m`.

**Parameters**

<code>__c</code>	The char to compare the mask of.
<code>__m</code>	The mask to compare against.

**Returns**

True if `__m & table[__c]` is true, false otherwise.

Definition at line 43 of file `ctype_inline.h`.

**4.540.3.11** `const char * std::ctype< char >::is ( const char * __lo, const char * __hi, mask * __vec ) const` `[inline],[inherited]`

Return a mask array.

This function finds the mask for each char in the range [lo, hi) and successively writes it to vec. vec must have as many elements as the char array.

**Parameters**

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.

**Returns**

`__hi`.

Definition at line 48 of file `ctype_inline.h`.

**4.540.3.12** `char std::ctype< char >::narrow ( char_type __c, char __dfault ) const` `[inline],[inherited]`

Narrow char.

This function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived ctype<char> facet, c will be returned unchanged.

This function works as if it returns `ctype<char>::do_narrow(c)`. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

<code>__c</code>	The char to convert.
<code>__dfault</code>	Char to return if conversion fails.

## Returns

The converted character.

Definition at line 923 of file locale\_facets.h.

References `std::ctype< _CharT >::do_narrow()`.

**4.540.3.13** `const char_type* std::ctype< char >::narrow ( const char_type * __lo, const char_type * __hi, char __dfault, char * __to ) const` `[inline]`, `[inherited]`

Narrow char array.

This function converts each char in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived `ctype<char>` facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_narrow(lo, hi, dfault, to)`. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__dfault</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

## Returns

`__hi`.

Definition at line 956 of file locale\_facets.h.

References `std::ctype< _CharT >::do_narrow()`.

**4.540.3.14** `const char * std::ctype< char >::scan_is ( mask __m, const char * __lo, const char * __hi ) const` `[inline]`, `[inherited]`

Find char matching a mask.

This function searches for and returns the first char in `[lo,hi)` for which `is(m,char)` is true.

## Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

## Returns

Pointer to a matching char if found, else `__hi`.

Definition at line 57 of file ctype\_inline.h.

**4.540.3.15** `const char * std::ctype< char >::scan_not ( mask __m, const char * __lo, const char * __hi ) const` [inline], [inherited]

Find char not matching a mask.

This function searches for and returns a pointer to the first char in [\_\_lo,\_\_hi) for which is(m,char) is false.

#### Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

#### Returns

Pointer to a non-matching char if found, else `__hi`.

Definition at line 67 of file `ctype_inline.h`.

**4.540.3.16** `const mask* std::ctype< char >::table ( ) const throw` [inline], [inherited]

Returns a pointer to the mask table provided to the constructor, or the default from `classic_table()` if none was provided.

Definition at line 974 of file `locale_facets.h`.

**4.540.3.17** `char_type std::ctype< char >::tolower ( char_type __c ) const` [inline], [inherited]

Convert to lowercase.

This function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`tolower()` acts as if it returns `ctype<char>::do_tolower(__c)`. `do_tolower()` must always return the same result for the same input.

#### Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

#### Returns

The lowercase char if convertible, else `__c`.

Definition at line 828 of file `locale_facets.h`.

References `std::ctype< _CharT >::do_tolower()`.

**4.540.3.18** `const char_type* std::ctype< char >::tolower ( char_type * __lo, const char_type * __hi ) const` [inline], [inherited]

Convert array to lowercase.

This function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

`tolower()` acts as if it returns `ctype<char>::do_tolower(__lo, __hi)`. `do_tolower()` must always return the same result for the same input.

## Parameters

<code>__lo</code>	Pointer to first char in range.
<code>__hi</code>	Pointer to end of range.

## Returns

`__hi`.

Definition at line 845 of file locale\_facets.h.

References std::ctype&lt; \_CharT &gt;::do\_tolower().

**4.540.3.19 char\_type std::ctype< char >::toupper ( char\_type \_\_c ) const** [inline],[inherited]

Convert to uppercase.

This function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

toupper() acts as if it returns ctype&lt;char&gt;::do\_toupper(c). do\_toupper() must always return the same result for the same input.

## Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

## Returns

The uppercase char if convertible, else `__c`.

Definition at line 795 of file locale\_facets.h.

References std::ctype&lt; \_CharT &gt;::do\_toupper().

**4.540.3.20 const char\_type\* std::ctype< char >::toupper ( char\_type \* \_\_lo, const char\_type \* \_\_hi ) const** [inline],[inherited]

Convert array to uppercase.

This function converts each char in the range [ \_\_lo, \_\_hi) to uppercase if possible. Other chars remain untouched.

toupper() acts as if it returns ctype&lt;char&gt;::do\_toupper(\_\_lo, \_\_hi). do\_toupper() must always return the same result for the same input.

## Parameters

<code>__lo</code>	Pointer to first char in range.
<code>__hi</code>	Pointer to end of range.

## Returns

`__hi`.

Definition at line 812 of file locale\_facets.h.

References std::ctype&lt; \_CharT &gt;::do\_toupper().

**4.540.3.21 char\_type std::ctype< char >::widen ( char \_\_c ) const** [inline],[inherited]

Widen char.

This function converts the `char` to `char_type` using the simplest reasonable transformation. For an undervied `ctype<char>` facet, the argument will be returned unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

#### Returns

The converted character.

Definition at line 865 of file `locale_facets.h`.

References `std::ctype<_CharT>::do_widen()`.

**4.540.3.22** `const char* std::ctype< char >::widen ( const char * __lo, const char * __hi, char_type * __to ) const`  
`[inline], [inherited]`

Widen char array.

This function converts each `char` in the input to `char` using the simplest reasonable transformation. For an undervied `ctype<char>` facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

<code>__lo</code>	Pointer to first char in range.
<code>__hi</code>	Pointer to end of range.
<code>__to</code>	Pointer to the destination array.

#### Returns

`__hi`.

Definition at line 892 of file `locale_facets.h`.

References `std::ctype<_CharT>::do_widen()`.

### 4.540.4 Member Data Documentation

**4.540.4.1** `locale::id std::ctype< char >::id` `[static], [inherited]`

The facet id for `ctype<char>`

Definition at line 696 of file `locale_facets.h`.

**4.540.4.2** `const size_t std::ctype< char >::table_size` `[static], [inherited]`

The size of the mask table. It is `SCHAR_MAX + 1`.

Definition at line 698 of file `locale_facets.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 4.541 `std::default_delete<_Tp>` Struct Template Reference

### Public Member Functions

- `template<typename _Up , typename = typename enable_if<is_convertible<_Up*, _Tp*>::value>::type> default_delete (const default_delete<_Up> &) noexcept`
- `void operator() (_Tp *__ptr) const`

#### 4.541.1 Detailed Description

`template<typename _Tp>struct std::default_delete<_Tp>`

Primary template, `default_delete`.

Definition at line 54 of file `unique_ptr.h`.

The documentation for this struct was generated from the following file:

- [unique\\_ptr.h](#)

## 4.542 `std::default_delete<_Tp[]>` Struct Template Reference

### Public Member Functions

- `template<typename _Up , typename = typename enable_if<!__is_derived_Tp<_Up>::value>::type> default_delete (const default_delete<_Up[]> &) noexcept`
- `void operator() (_Tp *__ptr) const`
- `template<typename _Up> enable_if<__is_derived_Tp<_Up>::value>::type operator() (_Up *) const =delete`

#### 4.542.1 Detailed Description

`template<typename _Tp>struct std::default_delete<_Tp[]>`

Specialization, `default_delete`.

Definition at line 75 of file `unique_ptr.h`.

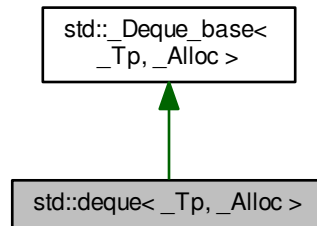
The documentation for this struct was generated from the following file:

- [unique\\_ptr.h](#)



#### 4.543 `std::deque<_Tp, _Alloc>` Class Template Reference

Inheritance diagram for `std::deque<_Tp, _Alloc>`:



#### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Base::const_iterator` **const\_iterator**
- typedef `_Tp_alloc_type::const_pointer` **const\_pointer**
- typedef `_Tp_alloc_type::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Tp_alloc_type::pointer` **pointer**
- typedef `_Tp_alloc_type::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

#### Public Member Functions

- `deque()`
- `deque(const allocator_type &__a)`
- `deque(size_type __n)`
- `deque(size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())`
- `deque(const deque &__x)`
- `deque(deque &&__x)`
- `deque(initializer_list< value_type > __l, const allocator_type &__a=allocator_type())`
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>> deque(_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())`
- `~deque()` noexcept
- `void assign(size_type __n, const value_type &__val)`
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>> void assign(_InputIterator __first, _InputIterator __last)`
- `void assign(initializer_list< value_type > __l)`

- reference [at](#) (size\_type \_\_n)
- const\_reference [at](#) (size\_type \_\_n) const
- reference [back](#) ()
- const\_reference [back](#) () const
- iterator [begin](#) () noexcept
- const\_iterator [begin](#) () const noexcept
- const\_iterator [cbegin](#) () const noexcept
- const\_iterator [cend](#) () const noexcept
- void [clear](#) () noexcept
- const\_reverse\_iterator [crbegin](#) () const noexcept
- const\_reverse\_iterator [crend](#) () const noexcept
- template<typename... \_Args> [iterator](#) [emplace](#) (iterator \_\_position, \_Args &&... \_\_args)
- template<typename... \_Args> void [emplace\\_back](#) (\_Args &&... \_\_args)
- template<typename... \_Args> void [emplace\\_front](#) (\_Args &&... \_\_args)
- bool [empty](#) () const noexcept
- iterator [end](#) () noexcept
- const\_iterator [end](#) () const noexcept
- iterator [erase](#) (iterator \_\_position)
- iterator [erase](#) (iterator \_\_first, iterator \_\_last)
- reference [front](#) ()
- const\_reference [front](#) () const
- allocator\_type [get\\_allocator](#) () const noexcept
- iterator [insert](#) (iterator \_\_position, const value\_type &\_\_x)
- iterator [insert](#) (iterator \_\_position, value\_type &&\_\_x)
- void [insert](#) (iterator \_\_p, initializer\_list< value\_type > \_\_l)
- void [insert](#) (iterator \_\_position, size\_type \_\_n, const value\_type &\_\_x)
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>> void [insert](#) (iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- size\_type [max\\_size](#) () const noexcept
- deque & operator= (const deque &\_\_x)
- deque & operator= (deque &&\_\_x)
- deque & operator= (initializer\_list< value\_type > \_\_l)
- reference [operator\[\]](#) (size\_type \_\_n)
- const\_reference [operator\[\]](#) (size\_type \_\_n) const
- void [pop\\_back](#) ()
- void [pop\\_front](#) ()
- void [push\\_back](#) (const value\_type &\_\_x)
- void [push\\_back](#) (value\_type &&\_\_x)
- void [push\\_front](#) (const value\_type &\_\_x)
- void [push\\_front](#) (value\_type &&\_\_x)
- reverse\_iterator [rbegin](#) () noexcept
- const\_reverse\_iterator [rbegin](#) () const noexcept
- reverse\_iterator [rend](#) () noexcept
- const\_reverse\_iterator [rend](#) () const noexcept
- void [resize](#) (size\_type \_\_new\_size)
- void [resize](#) (size\_type \_\_new\_size, const value\_type &\_\_x)
- void [shrink\\_to\\_fit](#) ()
- size\_type [size](#) () const noexcept
- void [swap](#) (deque &\_\_x)

## Protected Types

- enum { **\_S\_initial\_map\_size** }
- typedef `_Alloc::template rebind< _Tp * >::other` **\_Map\_alloc\_type**
- typedef pointer \* **\_Map\_pointer**

## Protected Member Functions

- `_Tp** _M_allocate_map (size_t __n)`
- `_Tp* _M_allocate_node ()`
- `template<typename _InputIterator > void _M_assign_aux (_InputIterator __first, _InputIterator __last, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator > void _M_assign_aux (_ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `template<typename _Integer > void _M_assign_dispatch (_Integer __n, _Integer __val, __true_type)`
- `template<typename _InputIterator > void _M_assign_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `void _M_create_nodes (_Tp** __nstart, _Tp** __nfinish)`
- `void _M_deallocate_map (_Tp** __p, size_t __n)`
- `void _M_deallocate_node (_Tp* __p)`
- `void _M_default_append (size_type __n)`
- `void _M_default_initialize ()`
- `template<typename _Alloc1 > void _M_destroy_data (iterator __first, iterator __last, const _Alloc1 &)`
- `void _M_destroy_data (iterator __first, iterator __last, const std::allocator< _Tp > &)`
- `void _M_destroy_data_aux (iterator __first, iterator __last)`
- `void _M_destroy_nodes (_Tp** __nstart, _Tp** __nfinish)`
- `void _M_erase_at_begin (iterator __pos)`
- `void _M_erase_at_end (iterator __pos)`
- `void _M_fill_assign (size_type __n, const value_type &__val)`
- `void \_M\_fill\_initialize (const value_type &__value)`
- `void _M_fill_insert (iterator __pos, size_type __n, const value_type &__x)`
- `_Map_alloc_type _M_get_map_allocator () const noexcept`
- `_Tp_alloc_type & _M_get_Tp_allocator () noexcept`
- `const _Tp_alloc_type & _M_get_Tp_allocator () const noexcept`
- `template<typename _Integer > void _M_initialize_dispatch (_Integer __n, _Integer __x, __true_type)`
- `template<typename _InputIterator > void _M_initialize_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `void \_M\_initialize\_map (size_t)`
- `template<typename... _Args> iterator _M_insert_aux (iterator __pos, _Args &&... __args)`
- `void _M_insert_aux (iterator __pos, size_type __n, const value_type &__x)`
- `template<typename _ForwardIterator > void _M_insert_aux (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, size_type __n)`
- `template<typename _Integer > void _M_insert_dispatch (iterator __pos, _Integer __n, _Integer __x, __true_type)`
- `template<typename _InputIterator > void _M_insert_dispatch (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)`
- `void \_M\_range\_check (size_type __n) const`
- `template<typename _InputIterator > void _M_range_insert_aux (iterator __pos, _InputIterator __first, _InputIterator __last, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator > void _M_range_insert_aux (iterator __pos, _ForwardIterator __first, \_ForwardIterator __last, std::forward\_iterator\_tag)`
- `bool _M_shrink_to_fit ()`

- `template<typename _InputIterator> void _M_range_initialize (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)`
- `template<typename _ForwardIterator> void _M_range_initialize (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
- `template<typename... _Args> void _M_push_back_aux (_Args &&... __args)`
- `template<typename... _Args> void _M_push_front_aux (_Args &&... __args)`
- `void _M_pop_back_aux ()`
- `void _M_pop_front_aux ()`
- `iterator _M_reserve_elements_at_front (size_type __n)`
- `iterator _M_reserve_elements_at_back (size_type __n)`
- `void _M_new_elements_at_front (size_type __new_elements)`
- `void _M_new_elements_at_back (size_type __new_elements)`
- `void _M_reserve_map_at_back (size_type __nodes_to_add=1)`
- `void _M_reserve_map_at_front (size_type __nodes_to_add=1)`
- `void _M_reallocate_map (size_type __nodes_to_add, bool __add_at_front)`

#### Static Protected Member Functions

- `static size_t _S_buffer_size ()`

#### Protected Attributes

- `_Deque_impl _M_impl`

#### 4.543.1 Detailed Description

`template<typename _Tp, typename _Alloc = std::allocator<_Tp>> class std::deque<_Tp, _Alloc>`

A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.

#### Template Parameters

<code>_Tp</code>	Type of element.
<code>_Alloc</code>	Allocator type, defaults to <code>allocator&lt;_Tp&gt;</code> .

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#).

In previous HP/SGI versions of deque, there was an extra template parameter so users could control the node size. This extension turned out to violate the C++ standard (it can be detected using template template parameters), and it was removed.

Here's how a `deque<Tp>` manages memory. Each deque has 4 members:

- `Tp** _M_map`
- `size_t _M_map_size`
- `iterator _M_start, _M_finish`

map\_size is at least 8. map is an array of map\_size pointers-to-nodes. (The name map has nothing to do with the std::map class, and **nodes** should not be confused with std::list's usage of *node*.)

A *node* has no specific type name as such, but it is referred to as *node* in this file. It is a simple array-of-Tp. If Tp is very large, there will be one Tp element per node (i.e., an *array* of one). For non-huge Tp's, node size is inversely related to Tp size: the larger the Tp, the fewer Tp's will fit in a node. The goal here is to keep the total size of a node relatively small and constant over different Tp's, to improve allocator efficiency.

Not every pointer in the map array will point to a node. If the initial number of elements in the deque is small, the /middle/ map pointers will be valid, and the ones at the edges will be unused. This same situation will arise as the map grows: available map pointers, if any, will be on the ends. As new nodes are created, only a subset of the map's pointers need to be copied *outward*.

Class invariants:

- For any nonsingular iterator i:
  - i.node points to a member of the map array. (Yes, you read that correctly: i.node does not actually point to a node.) The member of the map array is what actually points to the node.
  - i.first == \*(i.node) (This points to the node (first Tp element).)
  - i.last == i.first + node\_size
  - i.cur is a pointer in the range [i.first, i.last). NOTE: the implication of this is that i.cur is always a dereferenceable pointer, even if i is a past-the-end iterator.
- Start and Finish are always nonsingular iterators. NOTE: this means that an empty deque must have one node, a deque with <N elements (where N is the node buffer size) must have one node, a deque with N through (2N-1) elements must have two nodes, etc.
- For every node other than start.node and finish.node, every element in the node is an initialized object. If start.↔ node == finish.node, then [start.cur, finish.cur) are initialized objects, and the elements outside that range are uninitialized storage. Otherwise, [start.cur, start.last) and [finish.first, finish.cur) are initialized objects, and [start.↔ first, start.cur) and [finish.cur, finish.last) are uninitialized storage.
- [map, map + map\_size) is a valid, non-empty range.
- [start.node, finish.node] is a valid range contained within [map, map + map\_size).
- A pointer in the range [map, map + map\_size) points to an allocated node if and only if the pointer is in the range [start.node, finish.node].

Here's the magic: nothing in deque is **aware** of the discontinuous storage!

The memory setup and layout occurs in the parent, \_Base, and the iterator class is entirely responsible for *leaping* from one node to the next. All the implementation routines for deque itself work only through the start and finish iterators. This keeps the routines simple and sane, and we can use other standard algorithms as well.

Definition at line 730 of file stl\_deque.h.

#### 4.543.2 Constructor & Destructor Documentation

4.543.2.1 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque<_Tp, _Alloc>::deque ( )`  
`[inline]`

Default constructor creates no elements.

Definition at line 782 of file stl\_deque.h.

4.543.2.2 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque<_Tp, _Alloc>::deque ( const allocator_type &__a ) [inline], [explicit]`

Creates a deque with no elements.

## Parameters

<code>__a</code>	An allocator object.
------------------	----------------------

Definition at line 790 of file `stl_deque.h`.

4.543.2.3 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque<_Tp, _Alloc>::deque ( size_type __n ) [inline], [explicit]`

Creates a deque with default constructed elements.

## Parameters

<code>__n</code>	The number of elements to initially create.
------------------	---

This constructor fills the deque with *n* default constructed elements.

Definition at line 802 of file `stl_deque.h`.

4.543.2.4 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque<_Tp, _Alloc>::deque ( size_type __n, const value_type & __value, const allocator_type & __a = allocator_type() ) [inline]`

Creates a deque with copies of an exemplar element.

## Parameters

<code>__n</code>	The number of elements to initially create.
<code>__value</code>	An element to copy.
<code>__a</code>	An allocator.

This constructor fills the deque with `__n` copies of `__value`.

Definition at line 814 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::_M_fill_initialize()`.

4.543.2.5 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque<_Tp, _Alloc>::deque ( const deque<_Tp, _Alloc> & __x ) [inline]`

Deque copy constructor.

## Parameters

<code>__x</code>	A deque of identical element and allocator types.
------------------	---

The newly-created deque uses a copy of the allocation object used by `__x`.

Definition at line 841 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::begin()`, and `std::deque<_Tp, _Alloc>::end()`.

4.543.2.6 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque<_Tp, _Alloc>::deque ( deque<_Tp, _Alloc> && __x ) [inline]`

Deque move constructor.

## Parameters

<code>__x</code>	A deque of identical element and allocator types.
------------------	---

The newly-created deque contains the exact contents of `__x`. The contents of `__x` are a valid, but unspecified deque.

Definition at line 855 of file `stl_deque.h`.

4.543.2.7 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque<_Tp, _Alloc>::deque (`  
`initializer_list<value_type> __l, const allocator_type & __a = allocator_type() ) [inline]`

Builds a deque from an initializer list.



**Parameters**

<code>__l</code>	An initializer_list.
<code>__a</code>	An allocator object.

Create a deque consisting of copies of the elements in the initializer\_list `__l`.

This will call the element type's copy constructor N times (where N is `__l.size()`) and do no memory reallocation.

Definition at line 869 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::_M_range_initialize()`.

**4.543.2.8** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator, typename = std::RequireInputIter<_InputIterator>> std::deque<_Tp, _Alloc>::deque ( _InputIterator __first, _InputIterator __last, const allocator_type & __a = allocator_type() ) [inline]`

Builds a deque from a range.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__a</code>	An allocator object.

Create a deque consisting of copies of the elements from `[__first, __last)`.

If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor N times (where N is `distance(__first, __last)`) and do no memory reallocation. But if only input iterators are used, then this will do at most 2N calls to the copy constructor, and logN memory reallocations.

Definition at line 896 of file `stl_deque.h`.

**4.543.2.9** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque<_Tp, _Alloc>::~~deque ( ) [inline], [noexcept]`

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 917 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::begin()`, and `std::deque<_Tp, _Alloc>::end()`.

**4.543.3 Member Function Documentation**

**4.543.3.1** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque<_Tp, _Alloc>::_M_fill_initialize ( const value_type & __value ) [protected]`

Fills the deque with copies of `value`.

**Parameters**

<code>__value</code>	Initial value.
----------------------	----------------

**Returns**

Nothing.

**Precondition**

\_M\_start and \_M\_finish have already been initialized, but none of the deque's elements have yet been constructed.

This function is called only when the user provides an explicit size (with or without an explicit exemplar value).

Referenced by std::deque< \_Tp, \_Alloc >::deque().

4.543.3.2 `template<typename _Tp, typename _Alloc> void std::_Deque_base< _Tp, _Alloc >::_M_initialize_map( size_t __num_elements )` [protected], [inherited]

Layout storage.

**Parameters**

<code>__num_elements</code>	The count of T's for which to allocate space at first.
-----------------------------	--

**Returns**

Nothing.

The initial underlying memory layout is a bit complicated...

Definition at line 582 of file stl\_deque.h.

References std::max().

4.543.3.3 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::_M_new_elements_at_back( size_type __new_elements )` [protected]

Memory-handling helpers for the previous internal insert functions.

Referenced by std::deque< \_Tp, \_Alloc >::\_M\_reserve\_elements\_at\_back().

4.543.3.4 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::_M_new_elements_at_front( size_type __new_elements )` [protected]

Memory-handling helpers for the previous internal insert functions.

Referenced by std::deque< \_Tp, \_Alloc >::\_M\_reserve\_elements\_at\_front().

4.543.3.5 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::_M_pop_back_aux( )` [protected]

Helper functions for push\_\* and pop\_\*.

Referenced by std::deque< \_Tp, \_Alloc >::pop\_back().

4.543.3.6 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::_M_pop_front_aux( )` [protected]

Helper functions for push\_\* and pop\_\*.

Referenced by std::deque< \_Tp, \_Alloc >::pop\_front().

4.543.3.7 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename... _Args> void std::deque< _Tp, _Alloc >::_M_push_back_aux( _Args &&... __args )` [protected]

Helper functions for push\_\* and pop\_\*.

Referenced by std::deque< \_Tp, \_Alloc >::push\_back().

4.543.3.8 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename... _Args> void std::deque<_Tp, _Alloc>::_M_push_front_aux ( _Args &&... __args ) [protected]`

Helper functions for push\_\* and pop\_\*.

Referenced by `std::deque<_Tp, _Alloc>::push_front()`.

4.543.3.9 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque<_Tp, _Alloc>::_M_range_check ( size_type __n ) const [inline], [protected]`

Safety check used only from `at()`.

Definition at line 1265 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::size()`.

Referenced by `std::deque<_Tp, _Alloc>::at()`.

4.543.3.10 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator > void std::deque<_Tp, _Alloc>::_M_range_initialize ( _InputIterator __first, _InputIterator __last, std::input_iterator_tag ) [protected]`

Fills the deque with whatever is in `[first,last)`.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

Returns

Nothing.

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using `push_back` on each value from the iterator.

Referenced by `std::deque<_Tp, _Alloc>::deque()`.

4.543.3.11 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _ForwardIterator > void std::deque<_Tp, _Alloc>::_M_range_initialize ( _ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag ) [protected]`

Fills the deque with whatever is in `[first,last)`.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

Returns

Nothing.

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using `push_back` on each value from the iterator.

4.543.3.12 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque<_Tp, _Alloc>::_M_reallocate_map ( size_type __nodes_to_add, bool __add_at_front ) [protected]`

Memory-handling helpers for the major map.

Makes sure the `_M_map` has space for new nodes. Does not actually add the nodes. Can invalidate `_M_map` pointers. (And consequently, deque iterators.)

Referenced by `std::deque< _Tp, _Alloc >::_M_reserve_map_at_back()`, and `std::deque< _Tp, _Alloc >::_M_reserve_map_at_front()`.

**4.543.3.13** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque< _Tp, _Alloc >::_M_reserve_elements_at_back( size_type __n ) [inline], [protected]`

Memory-handling helpers for the previous internal insert functions.

Definition at line 1898 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_new_elements_at_back()`.

**4.543.3.14** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque< _Tp, _Alloc >::_M_reserve_elements_at_front( size_type __n ) [inline], [protected]`

Memory-handling helpers for the previous internal insert functions.

Definition at line 1888 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_new_elements_at_front()`.

**4.543.3.15** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::_M_reserve_map_at_back( size_type __nodes_to_add = 1 ) [inline], [protected]`

Memory-handling helpers for the major map.

Makes sure the `_M_map` has space for new nodes. Does not actually add the nodes. Can invalidate `_M_map` pointers. (And consequently, deque iterators.)

Definition at line 1924 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_reallocate_map()`.

**4.543.3.16** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::_M_reserve_map_at_front( size_type __nodes_to_add = 1 ) [inline], [protected]`

Memory-handling helpers for the major map.

Makes sure the `_M_map` has space for new nodes. Does not actually add the nodes. Can invalidate `_M_map` pointers. (And consequently, deque iterators.)

Definition at line 1932 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_reallocate_map()`.

**4.543.3.17** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::assign ( size_type __n, const value_type & __val ) [inline]`

Assigns a given value to a deque.

Parameters

<code>__n</code>	Number of elements to be assigned.
<code>__val</code>	Value to be assigned.

This function fills a deque with `n` copies of the given value. Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 978 of file `stl_deque.h`.

Referenced by `std::deque< _Tp, _Alloc >::assign()`, and `std::deque< _Tp, _Alloc >::operator=()`.

4.543.3.18 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator, typename  
= std::_RequireInputIter<_InputIterator>> void std::deque<_Tp, _Alloc>::assign ( _InputIterator __first,  
_InputIterator __last ) [inline]`

Assigns a range to a deque.

#### Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

This function fills a deque with copies of the elements in the range `[__first, __last)`.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 997 of file `stl_deque.h`.

4.543.3.19 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque<_Tp, _Alloc>::assign (   
initializer_list< value_type > __l ) [inline]`

Assigns an initializer list to a deque.

#### Parameters

<code>__l</code>	An <code>initializer_list</code> .
------------------	------------------------------------

This function fills a deque with copies of the elements in the `initializer_list __l`.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 1022 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::assign()`.

4.543.3.20 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::deque<_Tp, _Alloc>::at (   
size_type __n ) [inline]`

Provides access to the data contained in the deque.

#### Parameters

<code>__n</code>	The index of the element for which data should be accessed.
------------------	---

#### Returns

Read/write reference to data.

#### Exceptions

<code>std::out_of_range</code>	If <code>__n</code> is an invalid index.
--------------------------------	--

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws `out_of_range` if the check fails.

Definition at line 1284 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::_M_range_check()`.

4.543.3.21 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::deque<_Tp, _Alloc>::at (   
size_type __n ) const [inline]`

Provides access to the data contained in the deque.

## Parameters

<code>__n</code>	The index of the element for which data should be accessed.
------------------	---

## Returns

Read-only (constant) reference to data.

## Exceptions

<code>std::out_of_range</code>	If <code>__n</code> is an invalid index.
--------------------------------	--

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws `out_of_range` if the check fails.

Definition at line 1302 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_range_check()`.

**4.543.3.22** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::deque< _Tp, _Alloc >::back ( )`  
`[inline]`

Returns a read/write reference to the data at the last element of the deque.

Definition at line 1329 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::end()`.

**4.543.3.23** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::deque< _Tp, _Alloc >::back ( ) const` `[inline]`

Returns a read-only (constant) reference to the data at the last element of the deque.

Definition at line 1341 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::end()`.

**4.543.3.24** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque< _Tp, _Alloc >::begin ( )`  
`[inline], [noexcept]`

Returns a read/write iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1037 of file `stl_deque.h`.

Referenced by `std::deque< _Tp, _Alloc >::clear()`, `std::deque< _Tp, _Alloc >::deque()`, `std::deque< _Tp, _Alloc >::front()`, `std::operator==( )`, and `std::deque< _Tp, _Alloc >::~~deque()`.

**4.543.3.25** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::deque< _Tp, _Alloc >::begin ( ) const` `[inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1045 of file `stl_deque.h`.

**4.543.3.26** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::deque< _Tp, _Alloc >::cbegin ( ) const` `[inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1108 of file `stl_deque.h`.

**4.543.3.27** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::deque<_Tp, _Alloc>::cend ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1117 of file `stl_deque.h`.

**4.543.3.28** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque<_Tp, _Alloc>::clear ( ) [inline], [noexcept]`

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1616 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::begin()`.

Referenced by `std::deque<_Tp, _Alloc>::operator=()`.

**4.543.3.29** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::deque<_Tp, _Alloc>::crbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1126 of file `stl_deque.h`.

**4.543.3.30** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::deque<_Tp, _Alloc>::crend ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1135 of file `stl_deque.h`.

**4.543.3.31** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename... _Args> iterator std::deque<_Tp, _Alloc>::emplace ( iterator __position, _Args &&... __args )`

Inserts an object in deque before specified iterator.

#### Parameters

<code>__position</code>	An iterator into the deque.
<code>__args</code>	Arguments.

#### Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with `T(std::forward<Args>(args)...) before the specified location.`

Referenced by `std::deque<_Tp, _Alloc>::insert()`.

**4.543.3.32** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> bool std::deque<_Tp, _Alloc>::empty ( ) const [inline], [noexcept]`

Returns true if the deque is empty. (Thus `begin()` would equal `end()`.)

Definition at line 1228 of file `stl_deque.h`.

4.543.3.33 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque<_Tp, _Alloc>::end ( )`  
`[inline], [noexcept]`

Returns a read/write iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1054 of file `stl_deque.h`.

Referenced by `std::deque<_Tp, _Alloc>::back()`, `std::deque<_Tp, _Alloc>::deque()`, `std::operator==(, and std::deque<_Tp, _Alloc>::~~deque().`

4.543.3.34 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::deque<_Tp, _Alloc>::end ( ) const` `[inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1063 of file `stl_deque.h`.

4.543.3.35 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque<_Tp, _Alloc>::erase (`  
`iterator __position )`

Remove element at given position.

Parameters

<code>__position</code>	Iterator pointing to element to be erased.
-------------------------	--

Returns

An iterator pointing to the next element (or `end()`).

This function will erase the element at the given position and thus shorten the deque by one.

The user is cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

4.543.3.36 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque<_Tp, _Alloc>::erase (`  
`iterator __first, iterator __last )`

Remove a range of elements.

Parameters

<code>__first</code>	Iterator pointing to the first element to be erased.
<code>__last</code>	Iterator pointing to one past the last element to be erased.

Returns

An iterator pointing to the element pointed to by `last` prior to erasing (or `end()`).

This function will erase the elements in the range `[__first, __last)` and shorten the deque accordingly.

The user is cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

4.543.3.37 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::deque<_Tp, _Alloc>::front ( )`  
`[inline]`

Returns a read/write reference to the data at the first element of the deque.



Definition at line 1313 of file `std_deque.h`.

References `std::deque<_Tp, _Alloc>::begin()`.

**4.543.3.38** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::deque<_Tp, _Alloc>::front ( ) const [inline]`

Returns a read-only (constant) reference to the data at the first element of the deque.

Definition at line 1321 of file `std_deque.h`.

References `std::deque<_Tp, _Alloc>::begin()`.

**4.543.3.39** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> allocator_type std::deque<_Tp, _Alloc>::get_allocator ( ) const [inline], [noexcept]`

Get a copy of the memory allocation object.

Definition at line 1028 of file `std_deque.h`.

**4.543.3.40** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque<_Tp, _Alloc>::insert ( iterator __position, const value_type & __x )`

Inserts given value into deque before specified iterator.

Parameters

<code>__position</code>	An iterator into the deque.
<code>__x</code>	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location.

Referenced by `std::deque<_Tp, _Alloc>::insert()`, and `std::deque<_Tp, _Alloc>::resize()`.

**4.543.3.41** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque<_Tp, _Alloc>::insert ( iterator __position, value_type && __x ) [inline]`

Inserts given rvalue into deque before specified iterator.

Parameters

<code>__position</code>	An iterator into the deque.
<code>__x</code>	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location.

Definition at line 1492 of file `std_deque.h`.

References `std::deque<_Tp, _Alloc>::emplace()`, and `std::move()`.

**4.543.3.42** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque<_Tp, _Alloc>::insert ( iterator __p, initializer_list< value_type > __l ) [inline]`

Inserts an initializer list into the deque.

## Parameters

<code>__p</code>	An iterator into the deque.
<code>__l</code>	An initializer_list.

This function will insert copies of the data in the initializer\_list `__l` into the deque before the location specified by `__p`. This is known as *list insert*.

Definition at line 1505 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::insert()`.

4.543.3.43 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::insert ( iterator __position, size_type __n, const value_type & __x ) [inline]`

Inserts a number of copies of given data into the deque.

## Parameters

<code>__position</code>	An iterator into the deque.
<code>__n</code>	Number of elements to be inserted.
<code>__x</code>	Data to be inserted.

This function will insert a specified number of copies of the given data before the location specified by `__position`.

Definition at line 1519 of file `stl_deque.h`.

4.543.3.44 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>> void std::deque< _Tp, _Alloc >::insert ( iterator __position, _InputIterator __first, _InputIterator __last ) [inline]`

Inserts a range into the deque.

## Parameters

<code>__position</code>	An iterator into the deque.
<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

This function will insert copies of the data in the range `[__first,__last)` into the deque before the location specified by `__position`. This is known as *range insert*.

Definition at line 1536 of file `stl_deque.h`.

4.543.3.45 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::deque< _Tp, _Alloc >::max_size ( ) const [inline],[noexcept]`

Returns the `size()` of the largest possible deque.

Definition at line 1147 of file `stl_deque.h`.

4.543.3.46 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> deque& std::deque< _Tp, _Alloc >::operator= ( const deque< _Tp, _Alloc > & __x )`

Deque assignment operator.

## Parameters

<code>__x</code>	A deque of identical element and allocator types.
------------------	---

All the elements of `x` are copied, but unlike the copy constructor, the allocator object is not copied.

4.543.3.47 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> deque& std::deque<_Tp, _Alloc>::operator= ( deque<_Tp, _Alloc> && __x ) [inline]`

Deque move assignment operator.

## Parameters

<code>__x</code>	A deque of identical element and allocator types.
------------------	---

The contents of `__x` are moved into this deque (without copying). `__x` is a valid, but unspecified deque.

Definition at line 939 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::clear()`, and `std::deque< _Tp, _Alloc >::swap()`.

**4.543.3.48** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> deque& std::deque< _Tp, _Alloc >::operator= (initializer_list< value_type > __l) [inline]`

Assigns an initializer list to a deque.

## Parameters

<code>__l</code>	An <code>initializer_list</code> .
------------------	------------------------------------

This function fills a deque with copies of the elements in the `initializer_list __l`.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 960 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::assign()`.

**4.543.3.49** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::deque< _Tp, _Alloc >::operator[] (size_type __n) [inline]`

Subscript access to the data contained in the deque.

## Parameters

<code>__n</code>	The index of the element for which data should be accessed.
------------------	---

## Returns

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 1244 of file `stl_deque.h`.

**4.543.3.50** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::deque< _Tp, _Alloc >::operator[] (size_type __n) const [inline]`

Subscript access to the data contained in the deque.

## Parameters

<code>__n</code>	The index of the element for which data should be accessed.
------------------	---

## Returns

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 1259 of file `stl_deque.h`.

**4.543.3.51** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque<_Tp, _Alloc>::pop_back ( )`  
`[inline]`

Removes last element.

This is a typical stack operation. It shrinks the deque by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before `pop_back()` is called.

Definition at line 1442 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::_M_pop_back_aux()`.

**4.543.3.52** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque<_Tp, _Alloc>::pop_front ( )`  
`[inline]`

Removes first element.

This is a typical stack operation. It shrinks the deque by one.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop_front()` is called.

Definition at line 1421 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::_M_pop_front_aux()`.

**4.543.3.53** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque<_Tp, _Alloc>::push_back (`  
`const value_type &__x ) [inline]`

Add data to the end of the deque.

Parameters

<code>__x</code>	Data to be added.
------------------	-------------------

This is a typical stack operation. The function creates an element at the end of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.

Definition at line 1390 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::_M_push_back_aux()`.

**4.543.3.54** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque<_Tp, _Alloc>::push_front (`  
`const value_type &__x ) [inline]`

Add data to the front of the deque.

Parameters

<code>__x</code>	Data to be added.
------------------	-------------------

This is a typical stack operation. The function creates an element at the front of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.

Definition at line 1359 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::_M_push_front_aux()`.

**4.543.3.55** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reverse_iterator std::deque<_Tp, _Alloc>`  
`>::rbegin ( ) [inline], [noexcept]`

Returns a read/write reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1072 of file `stl_deque.h`.

4.543.3.56 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::deque< _Tp, _Alloc >::rbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1081 of file `stl_deque.h`.

4.543.3.57 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reverse_iterator std::deque< _Tp, _Alloc >::rend ( ) [inline], [noexcept]`

Returns a read/write reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1090 of file `stl_deque.h`.

4.543.3.58 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::deque< _Tp, _Alloc >::rend ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1099 of file `stl_deque.h`.

4.543.3.59 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::resize ( size_type __new_size ) [inline]`

Resizes the deque to the specified number of elements.

Parameters

<code>__new_size</code>	Number of elements the deque should contain.
-------------------------	--

This function will resize the deque to the specified number of elements. If the number is smaller than the deque's current size the deque is truncated, otherwise default constructed elements are appended.

Definition at line 1161 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::size()`.

4.543.3.60 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::resize ( size_type __new_size, const value_type & __x ) [inline]`

Resizes the deque to the specified number of elements.

Parameters

<code>__new_size</code>	Number of elements the deque should contain.
<code>__x</code>	Data with which new elements should be populated.

This function will resize the deque to the specified number of elements. If the number is smaller than the deque's current size the deque is truncated, otherwise the deque is extended and new elements are populated with given data.

Definition at line 1183 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::insert()`, and `std::deque< _Tp, _Alloc >::size()`.

4.543.3.61 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::shrink_to_fit ( ) [inline]`

A non-binding request to reduce memory use.

Definition at line 1219 of file `stl_deque.h`.

4.543.3.62 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::deque<_Tp, _Alloc>::size ( )`  
`const [inline], [noexcept]`

Returns the number of elements in the deque.

Definition at line 1142 of file `stl_deque.h`.

Referenced by `std::deque<_Tp, _Alloc>::_M_range_check()`, `std::operator==()`, and `std::deque<_Tp, _Alloc>::resize()`.

4.543.3.63 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque<_Tp, _Alloc>::swap (`  
`deque<_Tp, _Alloc> &__x ) [inline]`

Swaps data with another deque.

#### Parameters

<code>__x</code>	A deque of the same element and allocator types.
------------------	--

This exchanges the elements between two deques in constant time. (Four pointers, so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(d1,d2)` will feed to this function.

Definition at line 1596 of file `stl_deque.h`.

References `std::swap()`.

Referenced by `std::deque<_Tp, _Alloc>::operator=()`, and `std::swap()`.

The documentation for this class was generated from the following file:

- [stl\\_deque.h](#)

## 4.544 `std::discard_block_engine<_RandomNumberEngine, __p, __r>` Class Template Reference

### Public Types

- `typedef _RandomNumberEngine::result_type result_type`

### Public Member Functions

- `discard_block_engine ()`
- `discard_block_engine (const _RandomNumberEngine &__rng)`
- `discard_block_engine (_RandomNumberEngine &&__rng)`
- `discard_block_engine (result_type __s)`
- `template<typename _Sseq, typename = typename std::enable_if<!std::is_same<_Sseq, discard_block_engine>::value && !std::is_same<_Sseq, _RandomNumberEngine>::value>::type> discard_block_engine (_Sseq &__q)`
- `const _RandomNumberEngine &base () const noexcept`
- `void discard (unsigned long long __z)`
- `result_type operator() ()`
- `void seed ()`
- `void seed (result_type __s)`
- `template<typename _Sseq> void seed (_Sseq &__q)`

### Static Public Member Functions

- `static constexpr result_type max ()`
- `static constexpr result_type min ()`

## Static Public Attributes

- static constexpr size\_t **block\_size**
- static constexpr size\_t **used\_block**

## Friends

- template<typename \_RandomNumberEngine1, size\_t \_\_p1, size\_t \_\_r1, typename \_CharT, typename \_Traits> std::basic\_ostream<\_CharT, \_Traits> & **operator<<** (std::basic\_ostream<\_CharT, \_Traits> &\_\_os, const [std::discard\\_block\\_engine<\\_RandomNumberEngine1, \\_\\_p1, \\_\\_r1>](#) &\_\_x)
- bool **operator==** (const [discard\\_block\\_engine](#) &\_\_lhs, const [discard\\_block\\_engine](#) &\_\_rhs)
- template<typename \_RandomNumberEngine1, size\_t \_\_p1, size\_t \_\_r1, typename \_CharT, typename \_Traits> std::basic\_istream<\_CharT, \_Traits> & **operator>>** (std::basic\_istream<\_CharT, \_Traits> &\_\_is, [std::discard\\_block\\_engine<\\_RandomNumberEngine1, \\_\\_p1, \\_\\_r1>](#) &\_\_x)

## 4.544.1 Detailed Description

```
template<typename _RandomNumberEngine, size_t __p, size_t __r> class std::discard_block_engine<_RandomNumberEngine, __p, __r>
```

Produces random numbers from some base engine by discarding blocks of data.

0 <= \_\_r <= \_\_p

Definition at line 854 of file random.h.

## 4.544.2 Member Typedef Documentation

4.544.2.1 `template<typename _RandomNumberEngine, size_t __p, size_t __r> typedef _RandomNumberEngine::result_type std::discard_block_engine<_RandomNumberEngine, __p, __r>::result_type`

The type of the generated random value.

Definition at line 857 of file random.h.

## 4.544.3 Constructor &amp; Destructor Documentation

4.544.3.1 `template<typename _RandomNumberEngine, size_t __p, size_t __r> std::discard_block_engine<_RandomNumberEngine, __p, __r>::discard_block_engine( ) [inline]`

Constructs a default discard\_block\_engine engine.

The underlying engine is default constructed as well.

Definition at line 872 of file random.h.

4.544.3.2 `template<typename _RandomNumberEngine, size_t __p, size_t __r> std::discard_block_engine<_RandomNumberEngine, __p, __r>::discard_block_engine( const _RandomNumberEngine &__rng ) [inline], [explicit]`

Copy constructs a discard\_block\_engine engine.

Copies an existing base class random number generator.



## Parameters

<code>__rng</code>	An existing (base class) engine object.
--------------------	---

Definition at line 882 of file random.h.

```
4.544.3.3  template<typename _RandomNumberEngine, size_t __p, size_t __r> std::discard_block_engine<
            _RandomNumberEngine, __p, __r>::discard_block_engine ( _RandomNumberEngine && __rng ) [inline],
            [explicit]
```

Move constructs a discard\_block\_engine engine.

Copies an existing base class random number generator.

## Parameters

<code>__rng</code>	An existing (base class) engine object.
--------------------	---

Definition at line 892 of file random.h.

```
4.544.3.4  template<typename _RandomNumberEngine, size_t __p, size_t __r> std::discard_block_engine<
            _RandomNumberEngine, __p, __r>::discard_block_engine ( result_type __s ) [inline], [explicit]
```

Seed constructs a discard\_block\_engine engine.

Constructs the underlying generator engine seeded with `__s`.

## Parameters

<code>__s</code>	A seed value for the base class engine.
------------------	---

Definition at line 902 of file random.h.

```
4.544.3.5  template<typename _RandomNumberEngine, size_t __p, size_t __r> template<typename _Sseq, typename
            = typename std::enable_if<!std::is_same<_Sseq, discard_block_engine>::value && !std::is_same<_Sseq,
            _RandomNumberEngine>::value> ::type> std::discard_block_engine< _RandomNumberEngine, __p, __r
            >::discard_block_engine ( _Sseq & __q ) [inline], [explicit]
```

Generator construct a discard\_block\_engine engine.

## Parameters

<code>__q</code>	A seed sequence.
------------------	------------------

Definition at line 915 of file random.h.

## 4.544.4 Member Function Documentation

```
4.544.4.1  template<typename _RandomNumberEngine, size_t __p, size_t __r> const _RandomNumberEngine&
            std::discard_block_engine< _RandomNumberEngine, __p, __r >::base ( ) const [inline], [noexcept]
```

Gets a const reference to the underlying generator engine object.

Definition at line 959 of file random.h.

```
4.544.4.2  template<typename _RandomNumberEngine, size_t __p, size_t __r> void std::discard_block_engine<
            _RandomNumberEngine, __p, __r>::discard ( unsigned long long __z ) [inline]
```

Discard a sequence of random numbers.

Definition at line 980 of file random.h.

4.544.4.3 `template<typename _RandomNumberEngine, size_t __p, size_t __r> static constexpr result_type  
std::discard_block_engine<_RandomNumberEngine, __p, __r>::max( ) [inline], [static]`

Gets the maximum value in the generated random number range.

Definition at line 973 of file random.h.

References std::max().

4.544.4.4 `template<typename _RandomNumberEngine, size_t __p, size_t __r> static constexpr result_type  
std::discard_block_engine<_RandomNumberEngine, __p, __r>::min( ) [inline], [static]`

Gets the minimum value in the generated random number range.

Definition at line 966 of file random.h.

References std::min().

4.544.4.5 `template<typename _RandomNumberEngine, size_t __p, size_t __r> result_type std::discard_block_engine<  
_RandomNumberEngine, __p, __r>::operator()( )`

Gets the next value in the generated random number sequence.

4.544.4.6 `template<typename _RandomNumberEngine, size_t __p, size_t __r> void std::discard_block_engine<  
_RandomNumberEngine, __p, __r>::seed( ) [inline]`

Reseeds the discard\_block\_engine object with the default seed for the underlying base class generator engine.

Definition at line 924 of file random.h.

4.544.4.7 `template<typename _RandomNumberEngine, size_t __p, size_t __r> void std::discard_block_engine<  
_RandomNumberEngine, __p, __r>::seed( result_type __s ) [inline]`

Reseeds the discard\_block\_engine object with the default seed for the underlying base class generator engine.

Definition at line 935 of file random.h.

4.544.4.8 `template<typename _RandomNumberEngine, size_t __p, size_t __r> template<typename _Sseq> void  
std::discard_block_engine<_RandomNumberEngine, __p, __r>::seed( _Sseq & __q ) [inline]`

Reseeds the discard\_block\_engine object with the given seed sequence.

Parameters

<code>__q</code>	A seed generator function.
------------------	----------------------------

Definition at line 948 of file random.h.

#### 4.544.5 Friends And Related Function Documentation

4.544.5.1 `template<typename _RandomNumberEngine, size_t __p, size_t __r> template<typename _RandomNumberEngine1,  
size_t __p1, size_t __r1, typename _CharT, typename _Traits> std::basic_ostream<_CharT, _Traits>& operator<<(  
std::basic_ostream<_CharT, _Traits> & __os, const std::discard_block_engine<_RandomNumberEngine1, __p1,  
__r1> & __x ) [friend]`

Inserts the current state of a discard\_block\_engine random number generator engine \_\_x into the output stream \_\_os.

## Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>discard_block_engine</code> random number generator engine.

## Returns

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

4.544.5.2 `template<typename _RandomNumberEngine, size_t __p, size_t __r> bool operator==( const discard_block_engine< _RandomNumberEngine, __p, __r > & __lhs, const discard_block_engine< _RandomNumberEngine, __p, __r > & __rhs ) [friend]`

Compares two `discard_block_engine` random number generator objects of the same type for equality.

## Parameters

<code>__lhs</code>	A <code>discard_block_engine</code> random number generator object.
<code>__rhs</code>	Another <code>discard_block_engine</code> random number generator object.

## Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 1004 of file `random.h`.

4.544.5.3 `template<typename _RandomNumberEngine, size_t __p, size_t __r> template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _Traits> std::basic_istream< _CharT, _Traits> & operator>> ( std::basic_istream< _CharT, _Traits> & __is, std::discard_block_engine< _RandomNumberEngine1, __p1, __r1 > & __x ) [friend]`

Extracts the current state of a `% subtract_with_carry_engine` random number generator engine `__x` from the input stream `__is`.

## Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>discard_block_engine</code> random number generator engine.

## Returns

The input stream with the state of `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

## 4.545 `std::discrete_distribution< _IntType >` Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef `_IntType` [result\\_type](#)

## Public Member Functions

- `template<typename _InputIterator > discrete_distribution (_InputIterator __wbegin, _InputIterator __wend)`
- `discrete_distribution (initializer_list< double > __wl)`
- `template<typename _Func > discrete_distribution (size_t __nw, double __xmin, double __xmax, _Func __fw)`
- `discrete_distribution (const param\_type &__p)`
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator > void __generate (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)`
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator > void __generate (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param\_type &__p)`
- `template<typename _UniformRandomNumberGenerator > void __generate (result\_type * __f, result\_type * __t, _UniformRandomNumberGenerator &__urng, const param\_type &__p)`
- `result\_type max () const`
- `result\_type min () const`
- `template<typename _UniformRandomNumberGenerator > result\_type operator() (_UniformRandomNumberGenerator &__urng)`
- `template<typename _UniformRandomNumberGenerator > result\_type operator() (_UniformRandomNumberGenerator &__urng, const param\_type &__p)`
- `param\_type param () const`
- `void param (const param\_type &__param)`
- `std::vector< double > probabilities () const`
- `void reset ()`

## Friends

- `template<typename _IntType1, typename _CharT, typename _Traits > std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::discrete_distribution< _IntType1 > &__x)`
- `bool operator== (const discrete\_distribution &__d1, const discrete\_distribution &__d2)`
- `template<typename _IntType1, typename _CharT, typename _Traits > std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::discrete_distribution< _IntType1 > &__x)`

## 4.545.1 Detailed Description

`template<typename _IntType = int> class std::discrete_distribution< _IntType >`

A discrete\_distribution random number distribution.

The formula for the discrete probability mass function is

Definition at line 5251 of file random.h.

## 4.545.2 Member Typedef Documentation

4.545.2.1 `template<typename _IntType = int> typedef _IntType std::discrete_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 5254 of file random.h.

#### 4.545.3 Member Function Documentation

**4.545.3.1** `template<typename _IntType = int> result_type std::discrete_distribution< _IntType >::max ( ) const`  
`[inline]`

Returns the least upper bound value of the distribution.

Definition at line 5371 of file random.h.

References `std::vector< _Tp, _Alloc >::empty()`, and `std::vector< _Tp, _Alloc >::size()`.

**4.545.3.2** `template<typename _IntType = int> result_type std::discrete_distribution< _IntType >::min ( ) const`  
`[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 5364 of file random.h.

**4.545.3.3** `template<typename _IntType = int> template<typename _UniformRandomNumberGenerator > result_type`  
`std::discrete_distribution< _IntType >::operator() ( _UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 5382 of file random.h.

**4.545.3.4** `template<typename _IntType = int> param_type std::discrete_distribution< _IntType >::param ( ) const`  
`[inline]`

Returns the parameter set of the distribution.

Definition at line 5349 of file random.h.

**4.545.3.5** `template<typename _IntType = int> void std::discrete_distribution< _IntType >::param ( const param_type &`  
`__param ) [inline]`

Sets the parameter set of the distribution.

##### Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 5357 of file random.h.

**4.545.3.6** `template<typename _IntType = int> std::vector<double> std::discrete_distribution< _IntType >::probabilities (`  
`) const [inline]`

Returns the probabilities of the distribution.

Definition at line 5339 of file random.h.

References `std::vector< _Tp, _Alloc >::empty()`.

**4.545.3.7** `template<typename _IntType = int> void std::discrete_distribution< _IntType >::reset ( ) [inline]`

Resets the distribution state.

Definition at line 5332 of file random.h.

#### 4.545.4 Friends And Related Function Documentation

4.545.4.1 `template<typename _IntType = int> template<typename _IntType1 , typename _CharT , typename _Traits >  
std::basic_ostream<_CharT, _Traits>& operator<< ( std::basic_ostream< _CharT, _Traits > &__os, const  
std::discrete_distribution<_IntType1 > &__x ) [friend]`

Inserts a discrete\_distribution random number distribution \_\_x into the output stream \_\_os.

## Parameters

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

## Returns

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

4.545.4.2 `template<typename _IntType = int> bool operator== ( const discrete_distribution< _IntType > &__d1, const discrete_distribution< _IntType > &__d2 ) [friend]`

Return true if two discrete distributions have the same parameters.

Definition at line 5417 of file `random.h`.

4.545.4.3 `template<typename _IntType = int> template<typename _IntType1, typename _CharT, typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<_CharT, _Traits > &__is, std::discrete_distribution<_IntType1 > &__x ) [friend]`

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

## Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>discrete_distribution</code> random number generator engine.

## Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

4.546 `std::discrete_distribution< _IntType >::param_type` Struct Reference

## Public Types

- typedef `discrete_distribution< _IntType >` **distribution\_type**

## Public Member Functions

- `template<typename _InputIterator > param_type (_InputIterator __wbegin, _InputIterator __wend)`
- `param_type (initializer_list< double > __wil)`
- `template<typename _Func > param_type (size_t __nw, double __xmin, double __xmax, _Func __fw)`
- `param_type (const param\_type &)=default`
- `param\_type & operator= (const param\_type &)=default`
- `std::vector< double > probabilities () const`

## Friends

- class `discrete_distribution< _IntType >`
- bool `operator== (const param\_type &__p1, const param\_type &__p2)`

## 4.546.1 Detailed Description

```
template<typename _IntType = int> struct std::discrete_distribution< _IntType >::param_type
```

Parameter type.

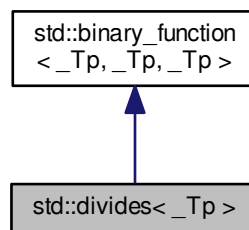
Definition at line 5260 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.547 `std::divides<_Tp>` Struct Template Reference

Inheritance diagram for `std::divides<_Tp>`:



## Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Tp` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

## Public Member Functions

- `_Tp` **operator()** (const `_Tp` &\_\_x, const `_Tp` &\_\_y) const

## 4.547.1 Detailed Description

```
template<typename _Tp> struct std::divides<_Tp>
```

One of the [math functors](#).

Definition at line 167 of file `stl_function.h`.



#### 4.547.2 Member Typedef Documentation

##### 4.547.2.1 `typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

##### 4.547.2.2 `typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

##### 4.547.2.3 `typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

#### 4.548 `std::enable_shared_from_this<_Tp>` Class Template Reference

##### Public Member Functions

- [shared\\_ptr<\\_Tp>](#) **shared\_from\_this** ()
- [shared\\_ptr<const \\_Tp>](#) **shared\_from\_this** () const

##### Protected Member Functions

- **enable\_shared\_from\_this** (const [enable\\_shared\\_from\\_this](#) &) noexcept
- **enable\_shared\_from\_this & operator=** (const [enable\\_shared\\_from\\_this](#) &) noexcept

##### Friends

- `template<typename _Tp1> void __enable_shared_from_this_helper (const __shared_count<> &__pn, const enable\_shared\_from\_this *__pe, const _Tp1 *__px) noexcept`

#### 4.548.1 Detailed Description

`template<typename _Tp> class std::enable_shared_from_this<_Tp>`

Base class allowing use of member function `shared_from_this`.

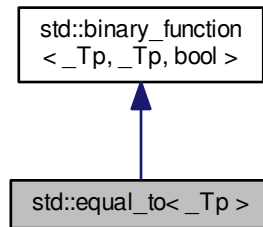
Definition at line 541 of file `shared_ptr.h`.

The documentation for this class was generated from the following file:

- [shared\\_ptr.h](#)

4.549 `std::equal_to<_Tp>` Struct Template Reference

Inheritance diagram for `std::equal_to<_Tp>`:



## Public Types

- `typedef _Tp` [first\\_argument\\_type](#)
- `typedef bool` [result\\_type](#)
- `typedef _Tp` [second\\_argument\\_type](#)

## Public Member Functions

- `bool` **`operator()`** (`const _Tp &__x, const _Tp &__y`) `const`

## 4.549.1 Detailed Description

`template<typename _Tp>struct std::equal_to<_Tp>`

One of the [comparison functors](#).

Definition at line 204 of file `stl_function.h`.

## 4.549.2 Member Typedef Documentation

4.549.2.1 `typedef _Tp` `std::binary_function<_Tp, _Tp, bool>::first_argument_type` `[inherited]`

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.549.2.2 `typedef bool` `std::binary_function<_Tp, _Tp, bool>::result_type` `[inherited]`

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.549.2.3 `typedef _Tp std::binary_function<_Tp, _Tp, bool>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.550 `std::exponential_distribution<_RealType>` Class Template Reference

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

### Public Member Functions

- [exponential\\_distribution](#) (const [result\\_type](#) &\_\_lambda=[result\\_type](#)(1))
- **`exponential_distribution`** (const [param\\_type](#) &\_\_p)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` > void **`__generate`** (`_ForwardIterator` \_\_f, `_ForwardIterator` \_\_t, `_UniformRandomNumberGenerator` &\_\_urng)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` > void **`__generate`** (`_ForwardIterator` \_\_f, `_ForwardIterator` \_\_t, `_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename `_UniformRandomNumberGenerator` > void **`__generate`** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, `_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- `_RealType` [lambda](#) () const
- [result\\_type](#) [max](#) () const
- [result\\_type](#) [min](#) () const
- template<typename `_UniformRandomNumberGenerator` > [result\\_type](#) **`operator()`** (`_UniformRandomNumberGenerator` &\_\_urng)
- template<typename `_UniformRandomNumberGenerator` > [result\\_type](#) **`operator()`** (`_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) [param](#) () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()

### Friends

- bool [operator==](#) (const [exponential\\_distribution](#) &\_\_d1, const [exponential\\_distribution](#) &\_\_d2)

### 4.550.1 Detailed Description

```
template<typename _RealType = double>class std::exponential_distribution<_RealType>
```

An exponential continuous distribution for random numbers.

Mean	$\frac{1}{\lambda}$
Median	$\frac{\ln 2}{\lambda}$
Mode	<i>zero</i>
Range	$[0, \infty]$
Standard Deviation	$\frac{1}{\lambda}$

Table 2: Distribution Statistics

The formula for the exponential probability density function is  $p(x|\lambda) = \lambda e^{-\lambda x}$ .

Definition at line 4646 of file random.h.

#### 4.550.2 Member Typedef Documentation

4.550.2.1 `template<typename _RealType = double> typedef _RealType std::exponential_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 4649 of file random.h.

#### 4.550.3 Constructor & Destructor Documentation

4.550.3.1 `template<typename _RealType = double> std::exponential_distribution<_RealType>::exponential_distribution ( const result_type & __lambda = result_type(1) ) [inline], [explicit]`

Constructs an exponential distribution with inverse scale parameter  $\lambda$ .

Definition at line 4684 of file random.h.

#### 4.550.4 Member Function Documentation

4.550.4.1 `template<typename _RealType = double> _RealType std::exponential_distribution<_RealType>::lambda ( ) const [inline]`

Returns the inverse scale parameter of the distribution.

Definition at line 4705 of file random.h.

4.550.4.2 `template<typename _RealType = double> result_type std::exponential_distribution<_RealType>::max ( ) const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 4734 of file random.h.

References `std::max()`.

4.550.4.3 `template<typename _RealType = double> result_type std::exponential_distribution<_RealType>::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4727 of file random.h.

4.550.4.4 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type  
std::exponential_distribution< _RealType >::operator() ( _UniformRandomNumberGenerator & __urng )  
[inline]`

Generating functions.

Definition at line 4742 of file random.h.

4.550.4.5 `template<typename _RealType = double> param_type std::exponential_distribution< _RealType >::param ( )  
const [inline]`

Returns the parameter set of the distribution.

Definition at line 4712 of file random.h.

4.550.4.6 `template<typename _RealType = double> void std::exponential_distribution< _RealType >::param ( const  
param_type & __param ) [inline]`

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 4720 of file random.h.

4.550.4.7 `template<typename _RealType = double> void std::exponential_distribution< _RealType >::reset ( )  
[inline]`

Resets the distribution state.

Has no effect on exponential distributions.

Definition at line 4699 of file random.h.

#### 4.550.5 Friends And Related Function Documentation

4.550.5.1 `template<typename _RealType = double> bool operator== ( const exponential_distribution< _RealType > & __d1,  
const exponential_distribution< _RealType > & __d2 ) [friend]`

Return true if two exponential distributions have the same parameters.

Definition at line 4782 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

#### 4.551 std::exponential\_distribution< \_RealType >::param\_type Struct Reference

Public Types

- typedef `exponential_distribution< _RealType >` **distribution\_type**

Public Member Functions

- `param_type` (`_RealType` \_\_lambda=`_RealType`(1))
- `_RealType` **lambda** () const

## Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

## 4.551.1 Detailed Description

```
template<typename _RealType = double> struct std::exponential_distribution<_RealType>::param_type
```

Parameter type.

Definition at line 4655 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 4.552 std::extreme\_value\_distribution&lt;\_RealType&gt; Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef \_RealType [result\\_type](#)

## Public Member Functions

- **extreme\_value\_distribution** (\_RealType \_\_a=\_RealType(0), \_RealType \_\_b=\_RealType(1))
- **extreme\_value\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator> void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator> void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator> void **\_\_generate** ([result\\_type](#) \* \_\_f, [result\\_type](#) \* \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- \_RealType **a** () const
- \_RealType **b** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator> [result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator> [result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

## Friends

- bool **operator==** (const [extreme\\_value\\_distribution](#) &\_\_d1, const [extreme\\_value\\_distribution](#) &\_\_d2)

## 4.552.1 Detailed Description

```
template<typename _RealType = double>class std::extreme_value_distribution< _RealType >
```

A `extreme_value_distribution` random number distribution.

The formula for the normal probability mass function is

$$p(x|a, b) = \frac{1}{b} \exp\left(\frac{a-x}{b} - \exp\left(\frac{a-x}{b}\right)\right)$$

Definition at line 5051 of file `random.h`.

## 4.552.2 Member Typedef Documentation

4.552.2.1 `template<typename _RealType = double> typedef _RealType std::extreme_value_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 5054 of file `random.h`.

## 4.552.3 Member Function Documentation

4.552.3.1 `template<typename _RealType = double> _RealType std::extreme_value_distribution< _RealType >::a ( ) const`  
[inline]

Return the  $a$  parameter of the distribution.

Definition at line 5109 of file `random.h`.

4.552.3.2 `template<typename _RealType = double> _RealType std::extreme_value_distribution< _RealType >::b ( ) const`  
[inline]

Return the  $b$  parameter of the distribution.

Definition at line 5116 of file `random.h`.

4.552.3.3 `template<typename _RealType = double> result_type std::extreme_value_distribution< _RealType >::max ( ) const` [inline]

Returns the least upper bound value of the distribution.

Definition at line 5145 of file `random.h`.

References `std::max()`.

4.552.3.4 `template<typename _RealType = double> result_type std::extreme_value_distribution< _RealType >::min ( ) const` [inline]

Returns the greatest lower bound value of the distribution.

Definition at line 5138 of file `random.h`.

4.552.3.5 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type std::extreme_value_distribution< _RealType >::operator() ( _UniformRandomNumberGenerator & __urng )`  
[inline]

Generating functions.

Definition at line 5153 of file random.h.

4.552.3.6 `template<typename _RealType = double> param_type std::extreme_value_distribution<_RealType>::param ( ) const [inline]`

Returns the parameter set of the distribution.

Definition at line 5123 of file random.h.

4.552.3.7 `template<typename _RealType = double> void std::extreme_value_distribution<_RealType>::param ( const param_type &__param ) [inline]`

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 5131 of file random.h.

4.552.3.8 `template<typename _RealType = double> void std::extreme_value_distribution<_RealType>::reset ( ) [inline]`

Resets the distribution state.

Definition at line 5102 of file random.h.

#### 4.552.4 Friends And Related Function Documentation

4.552.4.1 `template<typename _RealType = double> bool operator==( const extreme_value_distribution<_RealType> &__d1, const extreme_value_distribution<_RealType> &__d2 ) [friend]`

Return true if two extreme value distributions have the same parameters.

Definition at line 5188 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

#### 4.553 std::extreme\_value\_distribution<\_RealType>::param\_type Struct Reference

Public Types

- typedef [extreme\\_value\\_distribution<\\_RealType>](#) **distribution\_type**

Public Member Functions

- **param\_type** (`_RealType __a=_RealType(0), _RealType __b=_RealType(1)`)
- `_RealType a` () const
- `_RealType b` () const

Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)



## 4.553.1 Detailed Description

```
template<typename _RealType = double>struct std::extreme_value_distribution< _RealType >::param_type
```

Parameter type.

Definition at line 5060 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.554 `std::fisher_f_distribution< _RealType >` Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef `_RealType` [result\\_type](#)

## Public Member Functions

- **fisher\_f\_distribution** (`_RealType __m=_RealType(1), _RealType __n=_RealType(1)`)
- **fisher\_f\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` > void **\_\_generate** (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng`)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` > void **\_\_generate** (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- template<typename `_UniformRandomNumberGenerator` > void **\_\_generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, `_UniformRandomNumberGenerator &__urng`)
- template<typename `_UniformRandomNumberGenerator` > void **\_\_generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, `_UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- `_RealType m` () const
- [result\\_type max](#) () const
- [result\\_type min](#) () const
- `_RealType n` () const
- template<typename `_UniformRandomNumberGenerator` > [result\\_type operator\(\)](#) (`_UniformRandomNumberGenerator &__urng`)
- template<typename `_UniformRandomNumberGenerator` > [result\\_type operator\(\)](#) (`_UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- [param\\_type param](#) () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()

## Friends

- template<typename `_RealType1` , typename `_CharT` , typename `_Traits` > `std::basic_ostream< _CharT, _Traits > &operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::fisher\_f\_distribution< _RealType1 > &__x)`

- bool `operator==` (const `fisher_f_distribution` &\_\_d1, const `fisher_f_distribution` &\_\_d2)
- template<typename \_RealType1, typename \_CharT, typename \_Traits> std::basic\_istream<\_CharT, \_Traits> & `operator>>` (std::basic\_istream<\_CharT, \_Traits> &\_\_is, std::fisher\_f\_distribution<\_RealType1> &\_\_x)

#### 4.554.1 Detailed Description

template<typename \_RealType = double> class std::fisher\_f\_distribution<\_RealType>

A fisher\_f\_distribution random number distribution.

The formula for the normal probability mass function is

$$p(x|m, n) = \frac{\Gamma((m+n)/2)}{\Gamma(m/2)\Gamma(n/2)} \left(\frac{m}{n}\right)^{m/2} x^{(m/2)-1} \left(1 + \frac{mx}{n}\right)^{-(m+n)/2}$$

Definition at line 3130 of file random.h.

#### 4.554.2 Member Typedef Documentation

4.554.2.1 template<typename \_RealType = double> typedef \_RealType std::fisher\_f\_distribution<\_RealType>::result\_type

The type of the range of the distribution.

Definition at line 3133 of file random.h.

#### 4.554.3 Member Function Documentation

4.554.3.1 template<typename \_RealType = double> result\_type std::fisher\_f\_distribution<\_RealType>::max ( ) const  
[inline]

Returns the least upper bound value of the distribution.

Definition at line 3224 of file random.h.

References std::max().

4.554.3.2 template<typename \_RealType = double> result\_type std::fisher\_f\_distribution<\_RealType>::min ( ) const  
[inline]

Returns the greatest lower bound value of the distribution.

Definition at line 3217 of file random.h.

4.554.3.3 template<typename \_RealType = double> template<typename \_UniformRandomNumberGenerator> result\_type  
std::fisher\_f\_distribution<\_RealType>::operator() ( \_UniformRandomNumberGenerator &\_\_urng ) [inline]

Generating functions.

Definition at line 3232 of file random.h.

4.554.3.4 template<typename \_RealType = double> param\_type std::fisher\_f\_distribution<\_RealType>::param ( ) const  
[inline]

Returns the parameter set of the distribution.

Definition at line 3202 of file random.h.

4.554.3.5 `template<typename _RealType = double> void std::fisher_f_distribution<_RealType >::param ( const  
param_type & __param ) [inline]`

Sets the parameter set of the distribution.

## Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 3210 of file random.h.

4.554.3.6 `template<typename _RealType = double> void std::fisher_f_distribution<_RealType>::reset ( ) [inline]`

Resets the distribution state.

Definition at line 3181 of file random.h.

References `std::gamma_distribution<_RealType>::reset()`.

## 4.554.4 Friends And Related Function Documentation

4.554.4.1 `template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > std::basic_ostream<_CharT, _Traits>& operator<< ( std::basic_ostream<_CharT, _Traits> & __os, const std::fisher_f_distribution<_RealType1> & __x ) [friend]`

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

## Parameters

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

## Returns

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

4.554.4.2 `template<typename _RealType = double> bool operator==( const fisher_f_distribution<_RealType> & __d1, const fisher_f_distribution<_RealType> & __d2 ) [friend]`

Return true if two Fisher f distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3280 of file random.h.

4.554.4.3 `template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<_CharT, _Traits> & __is, std::fisher_f_distribution<_RealType1> & __x ) [friend]`

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

## Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>fisher_f_distribution</code> random number generator engine.

## Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

## 4.555 `std::fisher_f_distribution<_RealType>::param_type` Struct Reference

### Public Types

- typedef `fisher_f_distribution<_RealType>` **`distribution_type`**

### Public Member Functions

- **`param_type`** (`_RealType __m=_RealType(1), _RealType __n=_RealType(1)`)
- `_RealType m` () const
- `_RealType n` () const

### Friends

- bool **`operator==`** (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

#### 4.555.1 Detailed Description

`template<typename _RealType = double>struct std::fisher_f_distribution<_RealType>::param_type`

Parameter type.

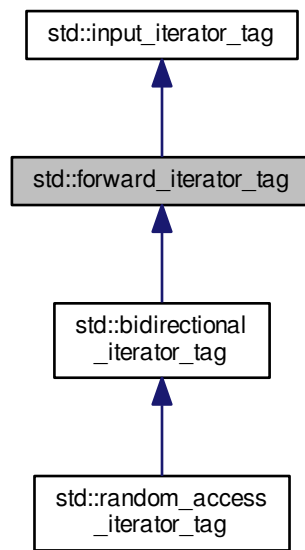
Definition at line 3139 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 4.556 std::forward\_iterator\_tag Struct Reference

Inheritance diagram for std::forward\_iterator\_tag:



### 4.556.1 Detailed Description

Forward iterators support a superset of input iterator operations.

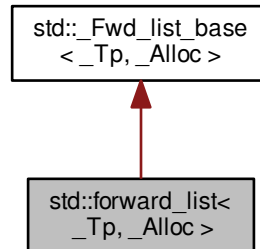
Definition at line 95 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

#### 4.557 std::forward\_list< \_Tp, \_Alloc > Class Template Reference

Inheritance diagram for std::forward\_list< \_Tp, \_Alloc >:



##### Public Types

- typedef \_Alloc **allocator\_type**
- typedef [\\_Fwd\\_list\\_const\\_iterator](#)< \_Tp > **const\_iterator**
- typedef \_Alloc\_traits::const\_pointer **const\_pointer**
- typedef \_Alloc\_traits::const\_reference **const\_reference**
- typedef std::ptrdiff\_t **difference\_type**
- typedef [\\_Fwd\\_list\\_iterator](#)< \_Tp > **iterator**
- typedef \_Alloc\_traits::pointer **pointer**
- typedef \_Alloc\_traits::reference **reference**
- typedef std::size\_t **size\_type**
- typedef \_Tp **value\_type**

##### Public Member Functions

- [forward\\_list](#) (const \_Alloc &\_\_al=\_Alloc())
- [forward\\_list](#) (const [forward\\_list](#) &\_\_list, const \_Alloc &\_\_al)
- [forward\\_list](#) ([forward\\_list](#) &&\_\_list, const \_Alloc &\_\_al) noexcept(\_Node\_alloc\_traits::\_S\_always\_equal())
- [forward\\_list](#) (size\_type \_\_n, const \_Alloc &\_\_al=\_Alloc())
- [forward\\_list](#) (size\_type \_\_n, const \_Tp &\_\_value, const \_Alloc &\_\_al=\_Alloc())
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>> [forward\\_list](#) (\_InputIterator \_\_first, \_↵  
\_InputIterator \_\_last, const \_Alloc &\_\_al=\_Alloc())
- [forward\\_list](#) (const [forward\\_list](#) &\_\_list)
- [forward\\_list](#) ([forward\\_list](#) &&\_\_list) noexcept
- [forward\\_list](#) (std::initializer\_list< \_Tp > \_\_il, const \_Alloc &\_\_al=\_Alloc())
- [~forward\\_list](#) () noexcept
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>> void [assign](#) (\_InputIterator \_\_first, \_↵  
\_InputIterator \_\_last)
- void [assign](#) (size\_type \_\_n, const \_Tp &\_\_val)
- void [assign](#) (std::initializer\_list< \_Tp > \_\_il)
- [iterator before\\_begin](#) () noexcept

- [const\\_iterator before\\_begin](#) () const noexcept
- [iterator begin](#) () noexcept
- [const\\_iterator begin](#) () const noexcept
- [const\\_iterator cbefore\\_begin](#) () const noexcept
- [const\\_iterator cbegin](#) () const noexcept
- [const\\_iterator cend](#) () const noexcept
- void [clear](#) () noexcept
- template<typename... \_Args> [iterator emplace\\_after](#) (const\_iterator \_\_pos, \_Args &&... \_\_args)
- template<typename... \_Args> void [emplace\\_front](#) (\_Args &&... \_\_args)
- bool [empty](#) () const noexcept
- [iterator end](#) () noexcept
- [const\\_iterator end](#) () const noexcept
- [iterator erase\\_after](#) (const\_iterator \_\_pos)
- [iterator erase\\_after](#) (const\_iterator \_\_pos, const\_iterator \_\_last)
- reference [front](#) ()
- const\_reference [front](#) () const
- allocator\_type [get\\_allocator](#) () const noexcept
- [iterator insert\\_after](#) (const\_iterator \_\_pos, const \_Tp &\_\_val)
- [iterator insert\\_after](#) (const\_iterator \_\_pos, \_Tp &&\_\_val)
- [iterator insert\\_after](#) (const\_iterator \_\_pos, size\_type \_\_n, const \_Tp &\_\_val)
- template<typename \_InputIterator, typename = std::RequireInputIter<\_InputIterator>> [iterator insert\\_after](#) (const\_iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last)
- [iterator insert\\_after](#) (const\_iterator \_\_pos, std::initializer\_list< \_Tp > \_\_il)
- size\_type [max\\_size](#) () const noexcept
- void [merge](#) (forward\_list &&\_\_list)
- void [merge](#) (forward\_list &\_\_list)
- template<typename \_Comp> void [merge](#) (forward\_list &&\_\_list, \_Comp \_\_comp)
- template<typename \_Comp> void [merge](#) (forward\_list &\_\_list, \_Comp \_\_comp)
- forward\_list & [operator=](#) (const forward\_list &\_\_list)
- forward\_list & [operator=](#) (forward\_list &&\_\_list) noexcept(\_Node\_alloc\_traits::\_S\_nothrow\_move())
- forward\_list & [operator=](#) (std::initializer\_list< \_Tp > \_\_il)
- void [pop\\_front](#) ()
- void [push\\_front](#) (const \_Tp &\_\_val)
- void [push\\_front](#) (\_Tp &&\_\_val)
- void [remove](#) (const \_Tp &\_\_val)
- template<typename \_Pred> void [remove\\_if](#) (\_Pred \_\_pred)
- void [resize](#) (size\_type \_\_sz)
- void [resize](#) (size\_type \_\_sz, const value\_type &\_\_val)
- void [reverse](#) () noexcept
- void [sort](#) ()
- template<typename \_Comp> void [sort](#) (\_Comp \_\_comp)
- void [splice\\_after](#) (const\_iterator \_\_pos, forward\_list &&\_\_list)
- void [splice\\_after](#) (const\_iterator \_\_pos, forward\_list &\_\_list)
- void [splice\\_after](#) (const\_iterator \_\_pos, forward\_list &&\_\_list, const\_iterator \_\_i)
- void [splice\\_after](#) (const\_iterator \_\_pos, forward\_list &\_\_list, const\_iterator \_\_i)
- void [splice\\_after](#) (const\_iterator \_\_pos, forward\_list &&, const\_iterator \_\_before, const\_iterator \_\_last)
- void [splice\\_after](#) (const\_iterator \_\_pos, forward\_list &, const\_iterator \_\_before, const\_iterator \_\_last)
- void [swap](#) (forward\_list &\_\_list) noexcept(\_Node\_alloc\_traits::\_S\_nothrow\_swap())
- void [unique](#) ()
- template<typename \_BinPred> void [unique](#) (\_BinPred \_\_binary\_pred)



### Private Member Functions

- `template<typename... _Args> \_Node * \_M\_create\_node ( \_Args &&...__args)`
- `\_Fwd\_list\_node\_base * \_M\_erase\_after ( \_Fwd\_list\_node\_base * __pos)`
- `\_Fwd\_list\_node\_base * \_M\_erase\_after ( \_Fwd\_list\_node\_base * __pos, \_Fwd\_list\_node\_base * __last)`
- `\_Node * \_M\_get\_node ()`
- `\_Node\_alloc\_type & \_M\_get\_Node\_allocator () noexcept`
- `const \_Node\_alloc\_type & \_M\_get\_Node\_allocator () const noexcept`
- `template<typename... _Args> \_Fwd\_list\_node\_base * \_M\_insert\_after (const_iterator __pos, \_Args &&...__args)`
- `void \_M\_put\_node ( \_Node * __p)`

### Private Attributes

- `\_Fwd\_list\_impl \_M\_impl`

### 4.557.1 Detailed Description

`template<typename \_Tp, typename \_Alloc = allocator<\_Tp>> class std::forward\_list<\_Tp, \_Alloc>`

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

#### Template Parameters

<code><a href="#">_Tp</a></code>	Type of element.
<code><a href="#">_Alloc</a></code>	Allocator type, defaults to <code><a href="#">allocator</a>&lt;<a href="#">_Tp</a>&gt;</code> .

Meets the requirements of a [container](#), a [sequence](#), including the [optional sequence requirements](#) with the exception of `at` and `operator[]`.

This is a *singly linked* list. Traversal up the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike `std::vector` and `std::deque`, random-access iterators are not provided, so subscripting ( `[]` ) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.

Also unlike the other standard containers, `std::forward_list` provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.

Definition at line 408 of file `forward_list.h`.

### 4.557.2 Constructor & Destructor Documentation

4.557.2.1 `template<typename \_Tp, typename \_Alloc = allocator<\_Tp>> std::forward\_list<\_Tp, \_Alloc>::forward_list ( const \_Alloc & __a/ = \_Alloc() ) [inline], [explicit]`

Creates a `forward_list` with no elements.

#### Parameters

<code>__a/</code>	An allocator object.
-------------------	----------------------

Definition at line 440 of file `forward_list.h`.

4.557.2.2 `template<typename \_Tp, typename \_Alloc = allocator<\_Tp>> std::forward\_list<\_Tp, \_Alloc>::forward_list ( const forward\_list<\_Tp, \_Alloc> & __list, const \_Alloc & __a/ ) [inline]`

Copy constructor with allocator argument.

## Parameters

<code>__list</code>	Input list to copy.
<code>__al</code>	An allocator object.

Definition at line 449 of file forward\_list.h.

References `std::forward_list< _Tp, _Alloc >::begin()`, and `std::forward_list< _Tp, _Alloc >::end()`.

4.557.2.3 `template<typename _Tp, typename _Alloc = allocator<_Tp>> std::forward_list< _Tp, _Alloc >::forward_list ( forward_list< _Tp, _Alloc > && __list, const _Alloc & __al ) [inline], [noexcept]`

Move constructor with allocator argument.

## Parameters

<code>__list</code>	Input list to move.
<code>__al</code>	An allocator object.

Definition at line 458 of file forward\_list.h.

4.557.2.4 `template<typename _Tp, typename _Alloc = allocator<_Tp>> std::forward_list< _Tp, _Alloc >::forward_list ( size_type __n, const _Alloc & __al = _Alloc() ) [inline], [explicit]`

Creates a forward\_list with default constructed elements.

## Parameters

<code>__n</code>	The number of elements to initially create.
------------------	---

This constructor creates the forward\_list with `__n` default constructed elements.

Definition at line 471 of file forward\_list.h.

4.557.2.5 `template<typename _Tp, typename _Alloc = allocator<_Tp>> std::forward_list< _Tp, _Alloc >::forward_list ( size_type __n, const _Tp & __value, const _Alloc & __al = _Alloc() ) [inline]`

Creates a forward\_list with copies of an exemplar element.

## Parameters

<code>__n</code>	The number of elements to initially create.
<code>__value</code>	An element to copy.
<code>__al</code>	An allocator object.

This constructor fills the forward\_list with `__n` copies of `__value`.

Definition at line 484 of file forward\_list.h.

4.557.2.6 `template<typename _Tp, typename _Alloc = allocator<_Tp>> template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>> std::forward_list< _Tp, _Alloc >::forward_list ( _InputIterator __first, _InputIterator __last, const _Alloc & __al = _Alloc() ) [inline]`

Builds a forward\_list from a range.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

<code>__al</code>	An allocator object.
-------------------	----------------------

Create a `forward_list` consisting of copies of the elements from `[__first,__last)`. This is linear in `N` (where `N` is `distance(__first,__last)`).

Definition at line 501 of file `forward_list.h`.

```
4.557.2.7 template<typename _Tp, typename _Alloc = allocator<_Tp>> std::forward_list<_Tp, _Alloc>::forward_list (
    const forward_list<_Tp, _Alloc> & __list ) [inline]
```

The `forward_list` copy constructor.

Parameters

<code>__list</code>	A <code>forward_list</code> of identical element and allocator types.
---------------------	---

Definition at line 511 of file `forward_list.h`.

References `std::forward_list<_Tp, _Alloc>::begin()`, and `std::forward_list<_Tp, _Alloc>::end()`.

```
4.557.2.8 template<typename _Tp, typename _Alloc = allocator<_Tp>> std::forward_list<_Tp, _Alloc>::forward_list (
    forward_list<_Tp, _Alloc> && __list ) [inline], [noexcept]
```

The `forward_list` move constructor.

Parameters

<code>__list</code>	A <code>forward_list</code> of identical element and allocator types.
---------------------	---

The newly-created `forward_list` contains the exact contents of `__list`. The contents of `__list` are a valid, but unspecified `forward_list`.

Definition at line 525 of file `forward_list.h`.

```
4.557.2.9 template<typename _Tp, typename _Alloc = allocator<_Tp>> std::forward_list<_Tp, _Alloc>::forward_list (
    std::initializer_list<_Tp> & __il, const _Alloc & __al = _Alloc() ) [inline]
```

Builds a `forward_list` from an `initializer_list`.

Parameters

<code>__il</code>	An <code>initializer_list</code> of <code>value_type</code> .
<code>__al</code>	An allocator object.

Create a `forward_list` consisting of copies of the elements in the `initializer_list` `__il`. This is linear in `__il.size()`.

Definition at line 536 of file `forward_list.h`.

```
4.557.2.10 template<typename _Tp, typename _Alloc = allocator<_Tp>> std::forward_list<_Tp, _Alloc>::~~forward_list (
    ) [inline], [noexcept]
```

The `forward_list` dtor.

Definition at line 544 of file `forward_list.h`.

#### 4.557.3 Member Function Documentation

```
4.557.3.1 template<typename _Tp, typename _Alloc = allocator<_Tp>> template<typename _InputIterator, typename =
    std::RequireInputIter<_InputIterator>> void std::forward_list<_Tp, _Alloc>::assign ( _InputIterator __first,
    _InputIterator __last ) [inline]
```

Assigns a range to a `forward_list`.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

This function fills a forward\_list with copies of the elements in the range [`__first`,`__last`).

Note that the assignment completely changes the forward\_list and that the number of elements of the resulting forward\_list is the same as the number of elements assigned. Old data is lost.

Definition at line 609 of file forward\_list.h.

Referenced by std::forward\_list< \_Tp, \_Alloc >::assign(), and std::forward\_list< \_Tp, \_Alloc >::operator=().

**4.557.3.2** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::assign ( size_type __n, const _Tp & __val ) [inline]`

Assigns a given value to a forward\_list.

## Parameters

<code>__n</code>	Number of elements to be assigned.
<code>__val</code>	Value to be assigned.

This function fills a forward\_list with `__n` copies of the given value. Note that the assignment completely changes the forward\_list, and that the resulting forward\_list has `__n` elements. Old data is lost.

Definition at line 626 of file forward\_list.h.

**4.557.3.3** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::assign ( std::initializer_list<_Tp> & __il ) [inline]`

Assigns an initializer\_list to a forward\_list.

## Parameters

<code>__il</code>	An initializer_list of value_type.
-------------------	------------------------------------

Replace the contents of the forward\_list with copies of the elements in the initializer\_list `__il`. This is linear in `il.size()`.

Definition at line 638 of file forward\_list.h.

References std::forward\_list< \_Tp, \_Alloc >::assign().

**4.557.3.4** `template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list< _Tp, _Alloc >::before_begin ( ) [inline], [noexcept]`

Returns a read/write iterator that points before the first element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 653 of file forward\_list.h.

**4.557.3.5** `template<typename _Tp, typename _Alloc = allocator<_Tp>> const_iterator std::forward_list< _Tp, _Alloc >::before_begin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points before the first element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 662 of file forward\_list.h.

**4.557.3.6** `template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list<_Tp, _Alloc>::begin ( )`  
`[inline], [noexcept]`

Returns a read/write iterator that points to the first element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 670 of file `forward_list.h`.

Referenced by `std::forward_list<_Tp, _Alloc>::forward_list()`.

**4.557.3.7** `template<typename _Tp, typename _Alloc = allocator<_Tp>> const_iterator std::forward_list<_Tp, _Alloc>::begin ( ) const` `[inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 679 of file `forward_list.h`.

**4.557.3.8** `template<typename _Tp, typename _Alloc = allocator<_Tp>> const_iterator std::forward_list<_Tp, _Alloc>::cbefore_begin ( ) const` `[inline], [noexcept]`

Returns a read-only (constant) iterator that points before the first element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 715 of file `forward_list.h`.

Referenced by `std::forward_list<_Tp, _Alloc>::emplace_front()`, and `std::forward_list<_Tp, _Alloc>::push_front()`.

**4.557.3.9** `template<typename _Tp, typename _Alloc = allocator<_Tp>> const_iterator std::forward_list<_Tp, _Alloc>::cbegin ( ) const` `[inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 706 of file `forward_list.h`.

**4.557.3.10** `template<typename _Tp, typename _Alloc = allocator<_Tp>> const_iterator std::forward_list<_Tp, _Alloc>::cend ( ) const` `[inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 724 of file `forward_list.h`.

**4.557.3.11** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list<_Tp, _Alloc>::clear ( )`  
`[inline], [noexcept]`

Erases all the elements.

Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1027 of file `forward_list.h`.

**4.557.3.12** `template<typename _Tp, typename _Alloc = allocator<_Tp>> template<typename... _Args> iterator`  
`std::forward_list<_Tp, _Alloc>::emplace_after ( const_iterator __pos, _Args &&... __args )` `[inline]`

Constructs object in `forward_list` after the specified iterator.

## Parameters

<code>__pos</code>	A const_iterator into the forward_list.
<code>__args</code>	Arguments.

## Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with `T(std::forward<Args>(args)...)`  after the specified location. Due to the nature of a forward\_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 837 of file forward\_list.h.

```
4.557.3.13  template<typename _Tp, typename _Alloc = allocator<_Tp>> template<typename... _Args> void
            std::forward_list< _Tp, _Alloc >::emplace_front ( _Args &&... __args ) [inline]
```

Constructs object in forward\_list at the front of the list.

## Parameters

<code>__args</code>	Arguments.
---------------------	------------

This function will insert an object of type Tp constructed with `Tp(std::forward<Args>(args)...)`  at the front of the list. Due to the nature of a forward\_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 781 of file forward\_list.h.

References `std::forward_list< _Tp, _Alloc >::cbefore_begin()`.

```
4.557.3.14  template<typename _Tp, typename _Alloc = allocator<_Tp>> bool std::forward_list< _Tp, _Alloc >::empty ( )
            const [inline],[noexcept]
```

Returns true if the forward\_list is empty. (Thus `begin()` would equal `end()`.)

Definition at line 732 of file forward\_list.h.

```
4.557.3.15  template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list< _Tp, _Alloc >::end ( )
            [inline],[noexcept]
```

Returns a read/write iterator that points one past the last element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 688 of file forward\_list.h.

Referenced by `std::forward_list< _Tp, _Alloc >::forward_list()`.

```
4.557.3.16  template<typename _Tp, typename _Alloc = allocator<_Tp>> const_iterator std::forward_list< _Tp, _Alloc
            >::end ( ) const [inline],[noexcept]
```

Returns a read-only iterator that points one past the last element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 697 of file forward\_list.h.

```
4.557.3.17  template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list< _Tp, _Alloc
            >::erase_after ( const_iterator __pos ) [inline]
```

Removes the element pointed to by the iterator following `pos`.

**Parameters**

<code>__pos</code>	Iterator pointing before element to be erased.
--------------------	--

**Returns**

An iterator pointing to the element following the one that was erased, or `end()` if no such element exists.

This function will erase the element at the given position and thus shorten the `forward_list` by one.

Due to the nature of a `forward_list` this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 940 of file `forward_list.h`.

**4.557.3.18** `template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list<_Tp, _Alloc>::erase_after ( const_iterator __pos, const_iterator __last ) [inline]`

Remove a range of elements.

**Parameters**

<code>__pos</code>	Iterator pointing before the first element to be erased.
<code>__last</code>	Iterator pointing to one past the last element to be erased.

**Returns**

`@ __last`.

This function will erase the elements in the range `(__pos, __last)` and shorten the `forward_list` accordingly.

This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 963 of file `forward_list.h`.

**4.557.3.19** `template<typename _Tp, typename _Alloc = allocator<_Tp>> reference std::forward_list<_Tp, _Alloc>::front ( ) [inline]`

Returns a read/write reference to the data at the first element of the `forward_list`.

Definition at line 749 of file `forward_list.h`.

**4.557.3.20** `template<typename _Tp, typename _Alloc = allocator<_Tp>> const_reference std::forward_list<_Tp, _Alloc>::front ( ) const [inline]`

Returns a read-only (constant) reference to the data at the first element of the `forward_list`.

Definition at line 760 of file `forward_list.h`.

**4.557.3.21** `template<typename _Tp, typename _Alloc = allocator<_Tp>> allocator_type std::forward_list<_Tp, _Alloc>::get_allocator ( ) const [inline], [noexcept]`

Get a copy of the memory allocation object.

Definition at line 643 of file `forward_list.h`.

4.557.3.22 `template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list<_Tp, _Alloc>::insert_after( const_iterator __pos, const _Tp & __val ) [inline]`

Inserts given value into forward\_list after specified iterator.



**Parameters**

<code>__pos</code>	An iterator into the <code>forward_list</code> .
<code>__val</code>	Data to be inserted.

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given value after the specified location. Due to the nature of a `forward_list` this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 854 of file `forward_list.h`.

Referenced by `std::forward_list<_Tp, _Alloc>::insert_after()`.

**4.557.3.23** `template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list<_Tp, _Alloc>::insert_after ( const_iterator __pos, size_type __n, const _Tp & __val )`

Inserts a number of copies of given data into the `forward_list`.

**Parameters**

<code>__pos</code>	An iterator into the <code>forward_list</code> .
<code>__n</code>	Number of elements to be inserted.
<code>__val</code>	Data to be inserted.

**Returns**

An iterator pointing to the last inserted copy of `val` or `pos` if `n == 0`.

This function will insert a specified number of copies of the given data after the location specified by `pos`.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

**4.557.3.24** `template<typename _Tp, typename _Alloc = allocator<_Tp>> template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>> iterator std::forward_list<_Tp, _Alloc>::insert_after ( const_iterator __pos, _InputIterator __first, _InputIterator __last )`

Inserts a range into the `forward_list`.

**Parameters**

<code>__pos</code>	An iterator into the <code>forward_list</code> .
<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

**Returns**

An iterator pointing to the last inserted element or `__pos` if `__first == __last`.

This function will insert copies of the data in the range `[__first, __last)` into the `forward_list` after the location specified by `__pos`.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

**4.557.3.25** `template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list<_Tp, _Alloc>::insert_after ( const_iterator __pos, std::initializer_list<_Tp> __il ) [inline]`

Inserts the contents of an `initializer_list` into `forward_list` after the specified iterator.

## Parameters

<code>__pos</code>	An iterator into the forward_list.
<code>__il</code>	An initializer_list of value_type.

## Returns

An iterator pointing to the last inserted element or `__pos` if `__il` is empty.

This function will insert copies of the data in the initializer\_list `__il` into the forward\_list before the location specified by `__pos`.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 919 of file forward\_list.h.

References std::forward\_list< \_Tp, \_Alloc >::insert\_after().

**4.557.3.26** `template<typename _Tp, typename _Alloc = allocator<_Tp>> size_type std::forward_list< _Tp, _Alloc >::max_size ( ) const [inline], [noexcept]`

Returns the largest possible number of elements of forward\_list.

Definition at line 739 of file forward\_list.h.

References std::allocator\_traits< \_Alloc >::max\_size().

**4.557.3.27** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::merge ( forward_list< _Tp, _Alloc > && __list ) [inline]`

Merge sorted lists.

## Parameters

<code>__list</code>	Sorted list to merge.
---------------------	-----------------------

Assumes that both `list` and this list are sorted according to operator<(). Merges elements of `__list` into this list in sorted order, leaving `__list` empty when complete. Elements in this list precede elements in `__list` that are equal.

Definition at line 1165 of file forward\_list.h.

References std::move().

**4.557.3.28** `template<typename _Tp, typename _Alloc = allocator<_Tp>> template<typename _Comp > void std::forward_list< _Tp, _Alloc >::merge ( forward_list< _Tp, _Alloc > && __list, _Comp __comp )`

Merge sorted lists according to comparison function.

## Parameters

<code>__list</code>	Sorted list to merge.
<code>__comp</code>	Comparison function defining sort order.

Assumes that both `__list` and this list are sorted according to comp. Merges elements of `__list` into this list in sorted order, leaving `__list` empty when complete. Elements in this list precede elements in `__list` that are equivalent according to comp().

**4.557.3.29** `template<typename _Tp, typename _Alloc = allocator<_Tp>> forward_list& std::forward_list< _Tp, _Alloc >::operator= ( const forward_list< _Tp, _Alloc > & __list )`

The forward\_list assignment operator.

## Parameters

<code>__list</code>	A <code>forward_list</code> of identical element and allocator types.
---------------------	---

All the elements of `__list` are copied, but unlike the copy constructor, the allocator object is not copied.

**4.557.3.30** `template<typename _Tp, typename _Alloc = allocator<_Tp>> forward_list& std::forward_list<_Tp, _Alloc>::operator= ( forward_list<_Tp, _Alloc> && __list ) [inline], [noexcept]`

The `forward_list` move assignment operator.

## Parameters

<code>__list</code>	A <code>forward_list</code> of identical element and allocator types.
---------------------	---

The contents of `__list` are moved into this `forward_list` (without copying, if the allocators permit it). `__list` is a valid, but unspecified `forward_list`

Definition at line 568 of file `forward_list.h`.

References `std::move()`.

**4.557.3.31** `template<typename _Tp, typename _Alloc = allocator<_Tp>> forward_list& std::forward_list<_Tp, _Alloc>::operator= ( std::initializer_list<_Tp> & __il ) [inline]`

The `forward_list` initializer list assignment operator.

## Parameters

<code>__il</code>	An <code>initializer_list</code> of <code>value_type</code> .
-------------------	---

Replace the contents of the `forward_list` with copies of the elements in the `initializer_list` `__il`. This is linear in `__il.size()`.

Definition at line 588 of file `forward_list.h`.

References `std::forward_list<_Tp, _Alloc>::assign()`.

**4.557.3.32** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list<_Tp, _Alloc>::pop_front ( ) [inline]`

Removes first element.

This is a typical stack operation. It shrinks the `forward_list` by one. Due to the nature of a `forward_list` this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop_front()` is called.

Definition at line 819 of file `forward_list.h`.

**4.557.3.33** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list<_Tp, _Alloc>::push_front ( const _Tp & __val ) [inline]`

Add data to the front of the `forward_list`.

## Parameters

<code>__val</code>	Data to be added.
--------------------	-------------------

This is a typical stack operation. The function creates an element at the front of the `forward_list` and assigns the given data to it. Due to the nature of a `forward_list` this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 796 of file `forward_list.h`.

References `std::forward_list<_Tp, _Alloc>::cbefore_begin()`.

```
4.557.3.34  template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list<_Tp, _Alloc >::remove (
    const _Tp & __val )
```

Remove all elements equal to value.

## Parameters

<code>__val</code>	The value to remove.
--------------------	----------------------

Removes every element in the list equal to `__val`. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**4.557.3.35** `template<typename _Tp, typename _Alloc = allocator<_Tp>> template<typename _Pred > void std::forward_list<_Tp, _Alloc>::remove_if ( _Pred __pred )`

Remove all elements satisfying a predicate.

## Parameters

<code>__pred</code>	Unary predicate function or object.
---------------------	-------------------------------------

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**4.557.3.36** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list<_Tp, _Alloc>::resize ( size_type __sz )`

Resizes the `forward_list` to the specified number of elements.

## Parameters

<code>__sz</code>	Number of elements the <code>forward_list</code> should contain.
-------------------	--

This function will resize the `forward_list` to the specified number of elements. If the number is smaller than the `forward_list`'s current number of elements the `forward_list` is truncated, otherwise the `forward_list` is extended and the new elements are default constructed.

**4.557.3.37** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list<_Tp, _Alloc>::resize ( size_type __sz, const value_type & __val )`

Resizes the `forward_list` to the specified number of elements.

## Parameters

<code>__sz</code>	Number of elements the <code>forward_list</code> should contain.
<code>__val</code>	Data with which new elements should be populated.

This function will resize the `forward_list` to the specified number of elements. If the number is smaller than the `forward_list`'s current number of elements the `forward_list` is truncated, otherwise the `forward_list` is extended and new elements are populated with given data.

**4.557.3.38** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list<_Tp, _Alloc>::reverse ( )`  
`[inline], [noexcept]`

Reverse the elements in list.

Reverse the order of elements in the list in linear time.

Definition at line 1218 of file `forward_list.h`.

**4.557.3.39** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list<_Tp, _Alloc>::sort ( )`  
`[inline]`

Sort the elements of the list.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 1199 of file forward\_list.h.

4.557.3.40 `template<typename _Tp, typename _Alloc = allocator<_Tp>> template<typename _Comp > void  
std::forward_list< _Tp, _Alloc >::sort ( _Comp __comp )`

Sort the forward\_list using a comparison function.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

4.557.3.41 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::splice_after (   
const_iterator __pos, forward_list< _Tp, _Alloc > && __list ) [inline]`

Insert contents of another forward\_list.

Parameters

<code>__pos</code>	Iterator referencing the element to insert after.
<code>__list</code>	Source list.

The elements of *list* are inserted in constant time after the element referenced by *pos*. *list* becomes an empty list.

Requires this != x.

Definition at line 1044 of file forward\_list.h.

4.557.3.42 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::splice_after (   
const_iterator __pos, forward_list< _Tp, _Alloc > && __list, const_iterator __i )`

Insert element from another forward\_list.

Parameters

<code>__pos</code>	Iterator referencing the element to insert after.
<code>__list</code>	Source list.
<code>__i</code>	Iterator referencing the element before the element to move.

Removes the element in list *list* referenced by *i* and inserts it into the current list after *pos*.

4.557.3.43 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::splice_after (   
const_iterator __pos, forward_list< _Tp, _Alloc > &&, const_iterator __before, const_iterator __last )  
[inline]`

Insert range from another forward\_list.

Parameters

<code>__pos</code>	Iterator referencing the element to insert after.
<code>__list</code>	Source list.
<code>__before</code>	Iterator referencing before the start of range in list.
<code>__last</code>	Iterator referencing the end of range in list.

Removes elements in the range ( \_\_before, \_\_last) and inserts them after \_\_pos in constant time.

Undefined if \_\_pos is in ( \_\_before, \_\_last).

Definition at line 1087 of file forward\_list.h.

4.557.3.44 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::swap (   
forward_list< _Tp, _Alloc > & __list ) [inline], [noexcept]`

Swaps data with another forward\_list.

## Parameters

<code>__list</code>	A forward_list of the same element and allocator types.
---------------------	---

This exchanges the elements between two lists in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(l1,l2)` will feed to this function.

Definition at line 980 of file `forward_list.h`.

References `std::swap()`.

Referenced by `std::swap()`.

**4.557.3.45** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list<_Tp, _Alloc>::unique ( )`  
`[inline]`

Remove consecutive duplicate elements.

For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1136 of file `forward_list.h`.

**4.557.3.46** `template<typename _Tp, typename _Alloc = allocator<_Tp>> template<typename _BinPred > void`  
`std::forward_list<_Tp, _Alloc>::unique ( _BinPred __binary_pred )`

Remove consecutive elements satisfying a predicate.

## Parameters

<code>__binary_pred</code>	Binary predicate function or object.
----------------------------	--------------------------------------

For each consecutive set of elements `[first,last)` that satisfy `predicate(first,i)` where `i` is an iterator in `[first,last)`, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

The documentation for this class was generated from the following file:

- [forward\\_list.h](#)

**4.558** `std::fpos<_StateT>` Class Template Reference

## Public Member Functions

- [fpos](#) ([streamoff](#) \_\_off)
- [operator streamoff](#) () const
- [fpos operator+](#) ([streamoff](#) \_\_off) const
- [fpos & operator+=](#) ([streamoff](#) \_\_off)
- [fpos operator-](#) ([streamoff](#) \_\_off) const
- [streamoff operator-](#) (const [fpos](#) &\_\_other) const
- [fpos & operator-=](#) ([streamoff](#) \_\_off)
- void [state](#) (\_StateT \_\_st)
- \_StateT [state](#) () const

**4.558.1** Detailed Description

```
template<typename _StateT>class std::fpos<_StateT>
```

Class representing stream positions.

The standard places no requirements upon the template parameter StateT. In this implementation StateT must be DefaultConstructible, CopyConstructible and Assignable. The standard only requires that fpos should contain a member of type StateT. In this implementation it also contains an offset stored as a signed integer.

#### Parameters

<i>StateT</i>	Type passed to and returned from state().
---------------	---

Definition at line 112 of file postypes.h.

### 4.558.2 Constructor & Destructor Documentation

4.558.2.1 `template<typename _StateT> std::fpos<_StateT>::fpos ( streamoff __off ) [inline]`

Construct position from offset.

Definition at line 133 of file postypes.h.

### 4.558.3 Member Function Documentation

4.558.3.1 `template<typename _StateT> std::fpos<_StateT>::operator streamoff ( ) const [inline]`

Convert to streamoff.

Definition at line 137 of file postypes.h.

4.558.3.2 `template<typename _StateT> fpos std::fpos<_StateT>::operator+ ( streamoff __off ) const [inline]`

Add position and offset.

Definition at line 178 of file postypes.h.

4.558.3.3 `template<typename _StateT> fpos& std::fpos<_StateT>::operator+= ( streamoff __off ) [inline]`

Add offset to this position.

Definition at line 154 of file postypes.h.

4.558.3.4 `template<typename _StateT> fpos std::fpos<_StateT>::operator- ( streamoff __off ) const [inline]`

Subtract offset from position.

Definition at line 192 of file postypes.h.

4.558.3.5 `template<typename _StateT> streamoff std::fpos<_StateT>::operator- ( const fpos<_StateT> &__other ) const [inline]`

Subtract position to return offset.

Definition at line 205 of file postypes.h.

4.558.3.6 `template<typename _StateT> fpos& std::fpos<_StateT>::operator-= ( streamoff __off ) [inline]`

Subtract offset from this position.

Definition at line 165 of file postypes.h.



4.558.3.7 `template<typename _StateT> void std::fpos<_StateT>::state ( _StateT __st ) [inline]`

Remember the value of *st*.

Definition at line 141 of file postypes.h.

4.558.3.8 `template<typename _StateT> _StateT std::fpos<_StateT>::state ( ) const [inline]`

Return the last set value of *st*.

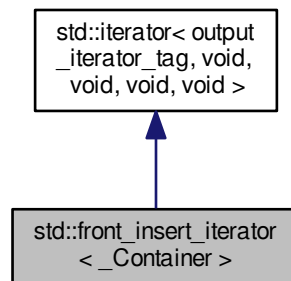
Definition at line 146 of file postypes.h.

The documentation for this class was generated from the following file:

- [postypes.h](#)

## 4.559 `std::front_insert_iterator<_Container>` Class Template Reference

Inheritance diagram for `std::front_insert_iterator<_Container>`:



### Public Types

- typedef `_Container` [container\\_type](#)
- typedef void [difference\\_type](#)
- typedef [output\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value\\_type](#)

### Public Member Functions

- [front\\_insert\\_iterator](#) (`_Container &__x`)
- [front\\_insert\\_iterator](#) & [operator\\*](#) ()
- [front\\_insert\\_iterator](#) & [operator++](#) ()
- [front\\_insert\\_iterator](#) [operator++](#) (int)
- [front\\_insert\\_iterator](#) & [operator=](#) (const typename `_Container::value_type` &\_\_value)
- [front\\_insert\\_iterator](#) & [operator=](#) (typename `_Container::value_type` &&\_\_value)

## Protected Attributes

- `_Container * container`

## 4.559.1 Detailed Description

```
template<typename _Container>class std::front_insert_iterator<_Container>
```

Turns assignment into insertion.

These are output iterators, constructed from a container-of-T. Assigning a T to the iterator prepends it to the container using `push_front`.

Tip: Using the `front_inserter` function to create these iterators can save typing.

Definition at line 493 of file `stl_iterator.h`.

## 4.559.2 Member Typedef Documentation

4.559.2.1 `template<typename _Container> typedef _Container std::front_insert_iterator<_Container>::container_type`

A nested typedef for the type of whatever container you used.

Definition at line 501 of file `stl_iterator.h`.

4.559.2.2 `typedef void std::iterator< output_iterator_tag, void, void, void, void>::difference_type [inherited]`

Distance between iterators is represented as this type.

Definition at line 125 of file `stl_iterator_base_types.h`.

4.559.2.3 `typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void>::iterator_category [inherited]`

One of the [tag types](#).

Definition at line 121 of file `stl_iterator_base_types.h`.

4.559.2.4 `typedef void std::iterator< output_iterator_tag, void, void, void, void>::pointer [inherited]`

This type represents a pointer-to-value\_type.

Definition at line 127 of file `stl_iterator_base_types.h`.

4.559.2.5 `typedef void std::iterator< output_iterator_tag, void, void, void, void>::reference [inherited]`

This type represents a reference-to-value\_type.

Definition at line 129 of file `stl_iterator_base_types.h`.

4.559.2.6 `typedef void std::iterator< output_iterator_tag, void, void, void, void>::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 123 of file `stl_iterator_base_types.h`.

## 4.559.3 Constructor &amp; Destructor Documentation

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

The only way to create this iterator is with a container.

Definition at line 504 of file `stl_iterator.h`.

#### 4.559.4 Member Function Documentation

4.559.4.1 `template<typename _Container > front_insert_iterator& std::front_insert_iterator< _Container >::operator* ( ) [inline]`

Simply returns `*this`.

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

4.559.4.2 `template<typename _Container > front_insert_iterator& std::front_insert_iterator< _Container >::operator++ ( ) [inline]`

Simply returns `*this`. (This iterator does not *move*.)

Definition at line 547 of file `stl_iterator.h`.

4.559.4.3 `template<typename _Container > front_insert_iterator std::front_insert_iterator< _Container >::operator++ ( int ) [inline]`

Simply returns `*this`. (This iterator does not *move*.)

Definition at line 552 of file `stl_iterator.h`.

4.559.4.4 `template<typename _Container > front_insert_iterator& std::front_insert_iterator< _Container >::operator= ( const typename _Container::value_type & __value ) [inline]`

#### Parameters

<code>__value</code>	An instance of whatever type <code>container_type::const_reference</code> is; presumably a reference-to- <code>const T</code> for <code>container&lt;T&gt;</code> .
----------------------	---

#### Returns

This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the front, if you like). Assigning a value to the iterator will always prepend the value to the front of the container.

Definition at line 526 of file `stl_iterator.h`.

The documentation for this class was generated from the following file:

- [stl\\_iterator.h](#)

## 4.560 `std::gamma_distribution< _RealType >` Class Template Reference

#### Classes

- struct [param\\_type](#)

## Public Types

- typedef \_RealType [result\\_type](#)

## Public Member Functions

- [gamma\\_distribution](#) (\_RealType \_\_alpha\_val=\_RealType(1), \_RealType \_\_beta\_val=\_RealType(1))
- [gamma\\_distribution](#) (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator> void [\\_\\_generate](#) (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator> void [\\_\\_generate](#) (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator> void [\\_\\_generate](#) ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- \_RealType [alpha](#) () const
- \_RealType [beta](#) () const
- [result\\_type](#) [max](#) () const
- [result\\_type](#) [min](#) () const
- template<typename \_UniformRandomNumberGenerator> [result\\_type](#) [operator\(\)](#) (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator> [result\\_type](#) [operator\(\)](#) (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) [param](#) () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()

## Friends

- template<typename \_RealType1, typename \_CharT, typename \_Traits> std::basic\_ostream<\_CharT, \_Traits> & [operator<<](#) (std::basic\_ostream<\_CharT, \_Traits> &\_\_os, const [std::gamma\\_distribution<\\_RealType1>](#) &\_\_x)
- bool [operator==](#) (const [gamma\\_distribution](#) &\_\_d1, const [gamma\\_distribution](#) &\_\_d2)
- template<typename \_RealType1, typename \_CharT, typename \_Traits> std::basic\_istream<\_CharT, \_Traits> & [operator>>](#) (std::basic\_istream<\_CharT, \_Traits> &\_\_is, [std::gamma\\_distribution<\\_RealType1>](#) &\_\_x)

## 4.560.1 Detailed Description

```
template<typename _RealType = double> class std::gamma_distribution<_RealType>
```

A gamma continuous distribution for random numbers.

The formula for the gamma probability density function is:

$$p(x|\alpha, \beta) = \frac{1}{\beta\Gamma(\alpha)} (x/\beta)^{\alpha-1} e^{-x/\beta}$$

Definition at line 2502 of file random.h.

#### 4.560.2 Member Typedef Documentation

4.560.2.1 `template<typename _RealType = double> typedef _RealType std::gamma_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 2505 of file random.h.

#### 4.560.3 Constructor & Destructor Documentation

4.560.3.1 `template<typename _RealType = double> std::gamma_distribution< _RealType >::gamma_distribution ( _RealType __alpha_val = _RealType(1), _RealType __beta_val = _RealType(1) ) [inline], [explicit]`

Constructs a gamma distribution with parameters  $\alpha$  and  $\beta$ .

Definition at line 2554 of file random.h.

#### 4.560.4 Member Function Documentation

4.560.4.1 `template<typename _RealType = double> _RealType std::gamma_distribution< _RealType >::alpha ( ) const [inline]`

Returns the  $\alpha$  of the distribution.

Definition at line 2575 of file random.h.

4.560.4.2 `template<typename _RealType = double> _RealType std::gamma_distribution< _RealType >::beta ( ) const [inline]`

Returns the  $\beta$  of the distribution.

Definition at line 2582 of file random.h.

4.560.4.3 `template<typename _RealType = double> result_type std::gamma_distribution< _RealType >::max ( ) const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 2611 of file random.h.

4.560.4.4 `template<typename _RealType = double> result_type std::gamma_distribution< _RealType >::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2604 of file random.h.

4.560.4.5 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator> result_type std::gamma_distribution< _RealType >::operator() ( _UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 2619 of file random.h.

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

4.560.4.6 `template<typename _RealType = double> param_type std::gamma_distribution<_RealType>::param ( ) const [inline]`

Returns the parameter set of the distribution.

Definition at line 2589 of file random.h.

4.560.4.7 `template<typename _RealType = double> void std::gamma_distribution<_RealType>::param ( const param_type & __param ) [inline]`

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 2597 of file random.h.

4.560.4.8 `template<typename _RealType = double> void std::gamma_distribution<_RealType>::reset ( ) [inline]`

Resets the distribution state.

Definition at line 2568 of file random.h.

Referenced by `std::chi_squared_distribution<_RealType>::reset()`, `std::fisher_f_distribution<_RealType>::reset()`, `std::student_t_distribution<_RealType>::reset()`, and `std::negative_binomial_distribution<_IntType>::reset()`.

#### 4.560.5 Friends And Related Function Documentation

4.560.5.1 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits> > std::basic_ostream<_CharT, _Traits>& operator<< ( std::basic_ostream<_CharT, _Traits> & __os, const std::gamma_distribution<_RealType1> & __x ) [friend]`

Inserts a gamma\_distribution random number distribution `__x` into the output stream `__os`.

Parameters

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

Returns

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

4.560.5.2 `template<typename _RealType = double> bool operator==( const gamma_distribution<_RealType> & __d1, const gamma_distribution<_RealType> & __d2 ) [friend]`

Return true if two gamma distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2655 of file random.h.

4.560.5.3 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits> > std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<_CharT, _Traits> & __is, std::gamma_distribution<_RealType1> & __x ) [friend]`

Extracts a gamma\_distribution random number distribution `__x` from the input stream `__is`.

## Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>gamma_distribution</code> random number generator engine.

## Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

4.561 `std::gamma_distribution<_RealType>::param_type` Struct Reference

## Public Types

- typedef [gamma\\_distribution](#)<\_RealType> **distribution\_type**

## Public Member Functions

- **param\_type** (\_RealType \_\_alpha\_val=\_RealType(1), \_RealType \_\_beta\_val=\_RealType(1))
- \_RealType **alpha** () const
- \_RealType **beta** () const

## Friends

- class **gamma\_distribution**<\_RealType>
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

## 4.561.1 Detailed Description

```
template<typename _RealType = double>struct std::gamma_distribution<_RealType>::param_type
```

Parameter type.

Definition at line 2511 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.562 `std::geometric_distribution<_IntType>` Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef \_IntType [result\\_type](#)

## Public Member Functions

- **geometric\_distribution** (double \_\_p=0.5)
- **geometric\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator> void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator> void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator> void **\_\_generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator> [result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator> [result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- double **p** () const
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

## Friends

- bool **operator==** (const [geometric\\_distribution](#) &\_\_d1, const [geometric\\_distribution](#) &\_\_d2)

## 4.562.1 Detailed Description

template<typename \_IntType = int> class std::geometric\_distribution<\_IntType>

A discrete geometric random number distribution.

The formula for the geometric probability density function is  $p(i|p) = p(1 - p)^i$  where  $p$  is the parameter of the distribution.

Definition at line 4008 of file random.h.

## 4.562.2 Member Typedef Documentation

4.562.2.1 template<typename \_IntType = int> typedef \_IntType std::geometric\_distribution<\_IntType>::result\_type

The type of the range of the distribution.

Definition at line 4011 of file random.h.

## 4.562.3 Member Function Documentation

4.562.3.1 template<typename \_IntType = int> [result\\_type](#) std::geometric\_distribution<\_IntType>::max ( ) const  
[inline]

Returns the least upper bound value of the distribution.

Definition at line 4100 of file random.h.

References std::max().



**4.562.3.2** `template<typename _IntType = int> result_type std::geometric_distribution<_IntType>::min ( ) const`  
`[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4093 of file random.h.

**4.562.3.3** `template<typename _IntType = int> template<typename _UniformRandomNumberGenerator> result_type`  
`std::geometric_distribution<_IntType>::operator() ( _UniformRandomNumberGenerator & __urng )`  
`[inline]`

Generating functions.

Definition at line 4108 of file random.h.

**4.562.3.4** `template<typename _IntType = int> double std::geometric_distribution<_IntType>::p ( ) const` `[inline]`

Returns the distribution parameter p.

Definition at line 4071 of file random.h.

**4.562.3.5** `template<typename _IntType = int> param_type std::geometric_distribution<_IntType>::param ( ) const`  
`[inline]`

Returns the parameter set of the distribution.

Definition at line 4078 of file random.h.

**4.562.3.6** `template<typename _IntType = int> void std::geometric_distribution<_IntType>::param ( const param_type &`  
`__param )` `[inline]`

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 4086 of file random.h.

**4.562.3.7** `template<typename _IntType = int> void std::geometric_distribution<_IntType>::reset ( )` `[inline]`

Resets the distribution state.

Does nothing for the geometric distribution.

Definition at line 4065 of file random.h.

#### 4.562.4 Friends And Related Function Documentation

**4.562.4.1** `template<typename _IntType = int> bool operator== ( const geometric_distribution<_IntType> & __d1, const`  
`geometric_distribution<_IntType> & __d2 )` `[friend]`

Return true if two geometric distributions have the same parameters.

Definition at line 4143 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

4.563 `std::geometric_distribution<_IntType>::param_type` Struct Reference

## Public Types

- typedef `geometric_distribution<_IntType>` **`distribution_type`**

## Public Member Functions

- **`param_type`** (double \_\_p=0.5)
- double **`p`** () const

## Friends

- class **`geometric_distribution<_IntType>`**
- bool **`operator==`** (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

## 4.563.1 Detailed Description

template<typename \_IntType = int>struct std::geometric\_distribution<\_IntType>::param\_type

Parameter type.

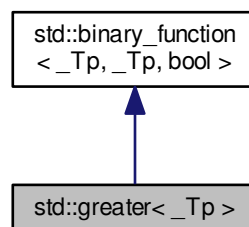
Definition at line 4017 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.564 `std::greater<_Tp>` Struct Template Reference

Inheritance diagram for `std::greater<_Tp>`:



## Public Types

- typedef `_Tp` `first_argument_type`
- typedef bool `result_type`
- typedef `_Tp` `second_argument_type`

## Public Member Functions

- **bool operator()** (const \_Tp &\_\_x, const \_Tp &\_\_y) const

## 4.564.1 Detailed Description

```
template<typename _Tp>struct std::greater< _Tp >
```

One of the [comparison functors](#).

Definition at line 222 of file `stl_function.h`.

## 4.564.2 Member Typedef Documentation

4.564.2.1 `typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.564.2.2 `typedef bool std::binary_function< _Tp, _Tp, bool >::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.564.2.3 `typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

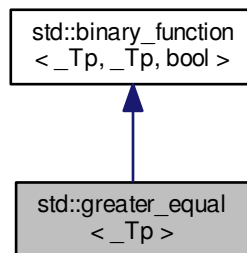
Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

4.565 `std::greater_equal< _Tp >` Struct Template Reference

Inheritance diagram for `std::greater_equal< _Tp >`:



## Public Types

- typedef [\\_Tp](#) [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef [\\_Tp](#) [second\\_argument\\_type](#)

## Public Member Functions

- bool **operator()** (const [\\_Tp](#) &\_\_x, const [\\_Tp](#) &\_\_y) const

## 4.565.1 Detailed Description

template<typename [\\_Tp](#)>struct std::greater\_equal< [\\_Tp](#) >

One of the [comparison functors](#).

Definition at line 240 of file [stl\\_function.h](#).

## 4.565.2 Member Typedef Documentation

4.565.2.1 typedef [\\_Tp](#) std::binary\_function< [\\_Tp](#), [\\_Tp](#), bool >::first\_argument\_type [inherited]

[first\\_argument\\_type](#) is the type of the first argument

Definition at line 117 of file [stl\\_function.h](#).

4.565.2.2 typedef bool std::binary\_function< [\\_Tp](#), [\\_Tp](#), bool >::result\_type [inherited]

[result\\_type](#) is the return type

Definition at line 123 of file [stl\\_function.h](#).

4.565.2.3 typedef [\\_Tp](#) std::binary\_function< [\\_Tp](#), [\\_Tp](#), bool >::second\_argument\_type [inherited]

[second\\_argument\\_type](#) is the type of the second argument

Definition at line 120 of file [stl\\_function.h](#).

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.566 std::gslice Class Reference

## Public Member Functions

- [gslice](#) ()
- [gslice](#) (size\_t \_\_o, const valarray< size\_t > &\_\_l, const valarray< size\_t > &\_\_s)
- [gslice](#) (const [gslice](#) &)
- [~gslice](#) ()
- [gslice](#) & operator= (const [gslice](#) &)
- valarray< size\_t > [size](#) () const
- size\_t [start](#) () const
- valarray< size\_t > [stride](#) () const

## Friends

- `template<typename _Tp> class valarray`

## 4.566.1 Detailed Description

Class defining multi-dimensional subset of an array.

The slice class represents a multi-dimensional subset of an array, specified by three parameter sets: start offset, size array, and stride array. The start offset is the index of the first element of the array that is part of the subset. The size and stride array describe each dimension of the slice. Size is the number of elements in that dimension, and stride is the distance in the array between successive elements in that dimension. Each dimension's size and stride is taken to begin at an array element described by the previous dimension. The size array and stride array must be the same size.

For example, if you have `offset==3`, `stride[0]==11`, `size[1]==3`, `stride[1]==3`, then `slice[0,0]==array[3]`, `slice[0,1]==array[6]`, `slice[0,2]==array[9]`, `slice[1,0]==array[14]`, `slice[1,1]==array[17]`, `slice[1,2]==array[20]`.

Definition at line 64 of file `gslice.h`.

The documentation for this class was generated from the following file:

- [gslice.h](#)

4.567 `std::gslice_array<_Tp>` Class Template Reference

## Public Types

- `typedef _Tp value_type`

## Public Member Functions

- [gslice\\_array](#) (const [gslice\\_array](#) &)
- void [operator%=\(const valarray<\\_Tp> &\)](#) const
- `template<class _Dom> void operator%=(const _Expr<_Dom, _Tp> &) const`
- void [operator&=\(const valarray<\\_Tp> &\)](#) const
- `template<class _Dom> void operator&=(const _Expr<_Dom, _Tp> &) const`
- void [operator\\*=\(const valarray<\\_Tp> &\)](#) const
- `template<class _Dom> void operator*=(const _Expr<_Dom, _Tp> &) const`
- void [operator+=\(const valarray<\\_Tp> &\)](#) const
- `template<class _Dom> void operator+=(const _Expr<_Dom, _Tp> &) const`
- void [operator-=\(const valarray<\\_Tp> &\)](#) const
- `template<class _Dom> void operator-=(const _Expr<_Dom, _Tp> &) const`
- void [operator/=\(const valarray<\\_Tp> &\)](#) const
- `template<class _Dom> void operator/=(const _Expr<_Dom, _Tp> &) const`
- void [operator<=<const valarray<\\_Tp> &\)](#) const
- `template<class _Dom> void operator<=<const _Expr<_Dom, _Tp> &) const`
- [gslice\\_array](#) & [operator=\(const gslice\\_array &\)](#)
- void [operator=\(const valarray<\\_Tp> &\)](#) const
- void [operator=\(const \\_Tp &\)](#) const
- `template<class _Dom> void operator=(const _Expr<_Dom, _Tp> &) const`
- void [operator>>=\(const valarray<\\_Tp> &\)](#) const
- `template<class _Dom> void operator>>=(const _Expr<_Dom, _Tp> &) const`

- void [operator^=](#) (const valarray< \_Tp > &) const
- template<class \_Dom > void **operator^=** (const \_Expr< \_Dom, \_Tp > &) const
- void [operator|=](#) (const valarray< \_Tp > &) const
- template<class \_Dom > void **operator|=** (const \_Expr< \_Dom, \_Tp > &) const

#### Friends

- class **valarray**< \_Tp >

#### 4.567.1 Detailed Description

template<typename \_Tp>class std::gslice\_array< \_Tp >

Reference to multi-dimensional subset of an array.

A gslice\_array is a reference to the actual elements of an array specified by a gslice. The way to get a gslice\_array is to call operator[](gslice) on a valarray. The returned gslice\_array then permits carrying operations out on the referenced subset of elements in the original valarray. For example, operator+=(valarray) will add values to the subset of elements in the underlying valarray this gslice\_array refers to.

#### Parameters

<i>Tp</i>	Element type.
-----------	---------------

Definition at line 60 of file gslice\_array.h.

The documentation for this class was generated from the following file:

- [gslice\\_array.h](#)

## 4.568 std::hash< \_Tp > Struct Template Reference

#### 4.568.1 Detailed Description

template<typename \_Tp>struct std::hash< \_Tp >

Primary class template hash.

Definition at line 58 of file functional\_hash.h.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 4.569 std::hash< \_\_gnu\_cxx::\_\_u16vstring > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

#### Public Types

- typedef \_Arg **argument\_type**
- typedef \_Result **result\_type**

## Public Member Functions

- `size_t operator()` (const [\\_\\_gnu\\_cxx::\\_\\_u16vstring](#) &\_\_s) const noexcept

### 4.569.1 Detailed Description

`template<> struct std::hash< \_\_gnu\_cxx::\_\_u16vstring >`

`std::hash` specialization for `__u16vstring`.

Definition at line 2826 of file `vstring.h`.

The documentation for this struct was generated from the following file:

- [vstring.h](#)

## 4.570 `std::hash< \_\_gnu\_cxx::\_\_u32vstring >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- `typedef _Arg argument_type`
- `typedef _Result result_type`

## Public Member Functions

- `size_t operator()` (const [\\_\\_gnu\\_cxx::\\_\\_u32vstring](#) &\_\_s) const noexcept

### 4.570.1 Detailed Description

`template<> struct std::hash< \_\_gnu\_cxx::\_\_u32vstring >`

`std::hash` specialization for `__u32vstring`.

Definition at line 2837 of file `vstring.h`.

The documentation for this struct was generated from the following file:

- [vstring.h](#)

## 4.571 `std::hash< \_\_gnu\_cxx::\_\_vstring >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- `typedef _Arg argument_type`
- `typedef _Result result_type`

## Public Member Functions

- `size_t operator()` (const [\\_\\_gnu\\_cxx::\\_\\_vstring](#) &\_\_s) const noexcept

## 4.571.1 Detailed Description

`template<> struct std::hash< __gnu_cxx::__vstring >`

`std::hash` specialization for `__vstring`.

Definition at line 2802 of file `vstring.h`.

The documentation for this struct was generated from the following file:

- [vstring.h](#)

4.572 `std::hash< __gnu_cxx::__wvstring >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- `typedef _Arg argument_type`
- `typedef _Result result_type`

## Public Member Functions

- `size_t operator()` (const [\\_\\_gnu\\_cxx::\\_\\_wvstring](#) &\_\_s) const noexcept

## 4.572.1 Detailed Description

`template<> struct std::hash< __gnu_cxx::__wvstring >`

`std::hash` specialization for `__wvstring`.

Definition at line 2813 of file `vstring.h`.

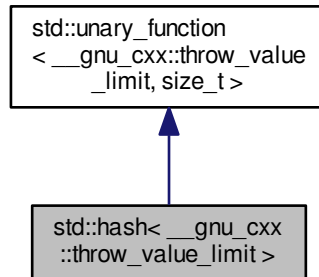
The documentation for this struct was generated from the following file:

- [vstring.h](#)



#### 4.573 `std::hash< __gnu_cxx::throw_value_limit >` Struct Template Reference

Inheritance diagram for `std::hash< __gnu_cxx::throw_value_limit >`:



##### Public Types

- typedef `__gnu_cxx::throw_value_limit` `argument_type`
- typedef `size_t` `result_type`

##### Public Member Functions

- `size_t operator()` (`const __gnu_cxx::throw_value_limit &__val`) `const`

##### 4.573.1 Detailed Description

`template<> struct std::hash< __gnu_cxx::throw_value_limit >`

Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.

Definition at line 788 of file `throw_allocator.h`.

##### 4.573.2 Member Typedef Documentation

**4.573.2.1** typedef `__gnu_cxx::throw_value_limit` `std::unary_function< __gnu_cxx::throw_value_limit, size_t >::argument_type` `[inherited]`

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

**4.573.2.2** typedef `size_t` `std::unary_function< __gnu_cxx::throw_value_limit, size_t >::result_type` `[inherited]`

`result_type` is the return type

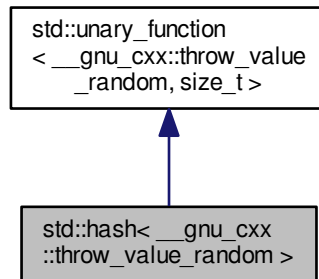
Definition at line 107 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

#### 4.574 `std::hash<__gnu_cxx::throw_value_random >` Struct Template Reference

Inheritance diagram for `std::hash<__gnu_cxx::throw_value_random >`:



##### Public Types

- typedef `__gnu_cxx::throw_value_random` `argument_type`
- typedef `size_t` `result_type`

##### Public Member Functions

- `size_t operator()` (const `__gnu_cxx::throw_value_random` &`__val`) const

##### 4.574.1 Detailed Description

```
template<> struct std::hash<__gnu_cxx::throw_value_random >
```

Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.

Definition at line 802 of file `throw_allocator.h`.

##### 4.574.2 Member Typedef Documentation

4.574.2.1 typedef `__gnu_cxx::throw_value_random` `std::unary_function<__gnu_cxx::throw_value_random, size_t>::argument_type` `[inherited]`

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

#### 4.574.2.2 `typedef size_t std::unary_function< __gnu_cxx::throw_value_random, size_t>::result_type` [inherited]

`result_type` is the return type

Definition at line 107 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

#### 4.575 `std::hash< __shared_ptr< _Tp, _Lp > >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

##### Public Types

- `typedef _Arg argument_type`
- `typedef _Result result_type`

##### Public Member Functions

- `size_t operator() (const __shared_ptr< _Tp, _Lp > &__s) const` noexcept

##### 4.575.1 Detailed Description

`template<typename _Tp, _Lock_policy _Lp>struct std::hash< __shared_ptr< _Tp, _Lp > >`

`std::hash` specialization for `__shared_ptr`.

Definition at line 1413 of file `shared_ptr_base.h`.

The documentation for this struct was generated from the following file:

- [shared\\_ptr\\_base.h](#)

#### 4.576 `std::hash< _Tp * >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

##### Public Types

- `typedef _Arg argument_type`
- `typedef _Result result_type`

##### Public Member Functions

- `size_t operator() (_Tp *__p) const` noexcept

## 4.576.1 Detailed Description

```
template<typename _Tp>struct std::hash< _Tp * >
```

Partial specializations for pointer types.

Definition at line 62 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

4.577 `std::hash< bool >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (`bool __val`) `const noexcept`

## 4.577.1 Detailed Description

```
template<>struct std::hash< bool >
```

Explicit specialization for `bool`.

Definition at line 80 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

4.578 `std::hash< char >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (`char __val`) `const noexcept`

#### 4.578.1 Detailed Description

`template<> struct std::hash< char >`

Explicit specialization for char.

Definition at line 83 of file functional\_hash.h.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

### 4.579 std::hash< char16\_t > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

#### Public Types

- typedef \_Arg **argument\_type**
- typedef \_Result **result\_type**

#### Public Member Functions

- `size_t operator() (char16_t __val) const` noexcept

#### 4.579.1 Detailed Description

`template<> struct std::hash< char16_t >`

Explicit specialization for char16\_t.

Definition at line 95 of file functional\_hash.h.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

### 4.580 std::hash< char32\_t > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

#### Public Types

- typedef \_Arg **argument\_type**
- typedef \_Result **result\_type**

#### Public Member Functions

- `size_t operator() (char32_t __val) const` noexcept

## 4.580.1 Detailed Description

`template<> struct std::hash< char32_t >`

Explicit specialization for `char32_t`.

Definition at line 98 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

4.581 `std::hash< double >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t` **operator()** (`double __val`) const noexcept

## 4.581.1 Detailed Description

`template<> struct std::hash< double >`

Specialization for `double`.

Definition at line 176 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

4.582 `std::hash< float >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t` **operator()** (`float __val`) const noexcept

#### 4.582.1 Detailed Description

`template<>struct std::hash< float >`

Specialization for float.

Definition at line 164 of file functional\_hash.h.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

### 4.583 std::hash< int > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

#### Public Types

- typedef \_Arg **argument\_type**
- typedef \_Result **result\_type**

#### Public Member Functions

- size\_t **operator()** (int \_\_val) const noexcept

#### 4.583.1 Detailed Description

`template<>struct std::hash< int >`

Explicit specialization for int.

Definition at line 104 of file functional\_hash.h.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

### 4.584 std::hash< long > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

#### Public Types

- typedef \_Arg **argument\_type**
- typedef \_Result **result\_type**

#### Public Member Functions

- size\_t **operator()** (long \_\_val) const noexcept

## 4.584.1 Detailed Description

`template<> struct std::hash< long >`

Explicit specialization for long.

Definition at line 107 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

4.585 `std::hash< long double >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t` **operator()** (`long double __val`) `const noexcept`

## 4.585.1 Detailed Description

`template<> struct std::hash< long double >`

Specialization for long double.

Definition at line 188 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

4.586 `std::hash< long long >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t` **operator()** (`long long __val`) `const noexcept`



#### 4.586.1 Detailed Description

`template<>struct std::hash< long long >`

Explicit specialization for long long.

Definition at line 110 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

#### 4.587 `std::hash< shared_ptr< _Tp > >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

##### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

##### Public Member Functions

- `size_t operator()` (const [shared\\_ptr](#)< `_Tp` > &\_\_s) const noexcept

#### 4.587.1 Detailed Description

`template<typename _Tp>struct std::hash< shared_ptr< _Tp > >`

`std::hash` specialization for `shared_ptr`.

Definition at line 619 of file `shared_ptr.h`.

The documentation for this struct was generated from the following file:

- [shared\\_ptr.h](#)

#### 4.588 `std::hash< short >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

##### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

##### Public Member Functions

- `size_t operator()` (short \_\_val) const noexcept

## 4.588.1 Detailed Description

`template<>struct std::hash< short >`

Explicit specialization for short.

Definition at line 101 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

4.589 `std::hash< signed char >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (`signed char __val`) `const noexcept`

## 4.589.1 Detailed Description

`template<>struct std::hash< signed char >`

Explicit specialization for signed char.

Definition at line 86 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

4.590 `std::hash< string >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (`const string &__s`) `const noexcept`

#### 4.590.1 Detailed Description

`template<>struct std::hash< string >`

`std::hash` specialization for `string`.

Definition at line 3044 of file `basic_string.h`.

The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

#### 4.591 `std::hash< u16string >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

##### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

##### Public Member Functions

- `size_t operator()` (const [u16string](#) &\_\_s) const noexcept

#### 4.591.1 Detailed Description

`template<>struct std::hash< u16string >`

`std::hash` specialization for `u16string`.

Definition at line 3077 of file `basic_string.h`.

The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

#### 4.592 `std::hash< u32string >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

##### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

##### Public Member Functions

- `size_t operator()` (const [u32string](#) &\_\_s) const noexcept

## 4.592.1 Detailed Description

`template<> struct std::hash< u32string >`

`std::hash` specialization for `u32string`.

Definition at line 3092 of file `basic_string.h`.

The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

4.593 `std::hash< unique_ptr< _Tp, _Dp > >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (const [unique\\_ptr](#)< `_Tp`, `_Dp` > &\_\_u) const noexcept

## 4.593.1 Detailed Description

`template<typename _Tp, typename _Dp> struct std::hash< unique_ptr< _Tp, _Dp > >`

`std::hash` specialization for `unique_ptr`.

Definition at line 599 of file `unique_ptr.h`.

The documentation for this struct was generated from the following file:

- [unique\\_ptr.h](#)

4.594 `std::hash< unsigned char >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (unsigned char \_\_val) const noexcept

#### 4.594.1 Detailed Description

`template<> struct std::hash< unsigned char >`

Explicit specialization for unsigned char.

Definition at line 89 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

#### 4.595 `std::hash< unsigned int >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

##### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

##### Public Member Functions

- `size_t operator()` (`unsigned int __val`) `const noexcept`

#### 4.595.1 Detailed Description

`template<> struct std::hash< unsigned int >`

Explicit specialization for unsigned int.

Definition at line 116 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

#### 4.596 `std::hash< unsigned long >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

##### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

##### Public Member Functions

- `size_t operator()` (`unsigned long __val`) `const noexcept`

## 4.596.1 Detailed Description

`template<> struct std::hash< unsigned long >`

Explicit specialization for unsigned long.

Definition at line 119 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

4.597 `std::hash< unsigned long long >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (`unsigned long long __val`) `const noexcept`

## 4.597.1 Detailed Description

`template<> struct std::hash< unsigned long long >`

Explicit specialization for unsigned long long.

Definition at line 122 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

4.598 `std::hash< unsigned short >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (`unsigned short __val`) `const noexcept`

#### 4.598.1 Detailed Description

`template<> struct std::hash< unsigned short >`

Explicit specialization for unsigned short.

Definition at line 113 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

### 4.599 `std::hash< wchar_t >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

#### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

#### Public Member Functions

- `size_t operator()` (`wchar_t __val`) `const noexcept`

#### 4.599.1 Detailed Description

`template<> struct std::hash< wchar_t >`

Explicit specialization for `wchar_t`.

Definition at line 92 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

### 4.600 `std::hash< wstring >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

#### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

#### Public Member Functions

- `size_t operator()` (`const wstring &__s`) `const noexcept`

## 4.600.1 Detailed Description

`template<> struct std::hash< wstring >`

`std::hash` specialization for `wstring`.

Definition at line 3059 of file `basic_string.h`.

The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

4.601 `std::hash<::vector< bool, _Alloc > >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (`const ::vector< bool, _Alloc > &`) `const` `noexcept`

## 4.601.1 Detailed Description

`template<typename _Alloc> struct std::hash<::vector< bool, _Alloc > >`

`std::hash` specialization for `vector<bool>`.

Definition at line 1143 of file `stl_bvector.h`.

The documentation for this struct was generated from the following file:

- [stl\\_bvector.h](#)

4.602 `std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >` Class Template Reference

## Public Types

- typedef `_UIntType` [result\\_type](#)

## Public Member Functions

- [independent\\_bits\\_engine](#) ()
- [independent\\_bits\\_engine](#) (`const _RandomNumberEngine &__rng`)
- [independent\\_bits\\_engine](#) (`_RandomNumberEngine &&__rng`)
- [independent\\_bits\\_engine](#) ([result\\_type](#) \_\_s)
- `template<typename _Sseq, typename = typename std::enable_if<!std::is_same<_Sseq, independent_bits_engine>::value && !std::is_<← same<_Sseq, _RandomNumberEngine>::value> ::type>` [independent\\_bits\\_engine](#) (`_Sseq &__q`)



- `const _RandomNumberEngine & base () const noexcept`
- `void discard (unsigned long long __z)`
- `result_type operator() ()`
- `void seed ()`
- `void seed (result_type __s)`
- `template<typename _Sseq > void seed (_Sseq &__q)`

#### Static Public Member Functions

- `static constexpr result_type max ()`
- `static constexpr result_type min ()`

#### Friends

- `bool operator== (const independent_bits_engine &__lhs, const independent_bits_engine &__rhs)`
- `template<typename _CharT, typename _Traits > std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)`

#### 4.602.1 Detailed Description

`template<typename _RandomNumberEngine, size_t __w, typename _UIntType> class std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >`

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits `__w`.

Definition at line 1074 of file random.h.

#### 4.602.2 Member Typedef Documentation

4.602.2.1 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType > typedef _UIntType std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::result_type`

The type of the generated random value.

Definition at line 1077 of file random.h.

#### 4.602.3 Constructor & Destructor Documentation

4.602.3.1 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType > std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine ( ) [inline]`

Constructs a default `independent_bits_engine` engine.

The underlying engine is default constructed as well.

Definition at line 1090 of file random.h.

```
4.602.3.2 template<typename _RandomNumberEngine, size_t __w, typename _UIntType > std::independent_bits_engine<
    _RandomNumberEngine, __w, _UIntType >::independent_bits_engine ( const _RandomNumberEngine & __rng )
    [inline], [explicit]
```

Copy constructs a independent\_bits\_engine engine.

Copies an existing base class random number generator.

Parameters

<code>__rng</code>	An existing (base class) engine object.
--------------------	---

Definition at line 1100 of file random.h.

```
4.602.3.3 template<typename _RandomNumberEngine, size_t __w, typename _UIntType > std::independent_bits_engine<
    _RandomNumberEngine, __w, _UIntType >::independent_bits_engine ( _RandomNumberEngine && __rng )
    [inline], [explicit]
```

Move constructs a independent\_bits\_engine engine.

Copies an existing base class random number generator.

Parameters

<code>__rng</code>	An existing (base class) engine object.
--------------------	---

Definition at line 1110 of file random.h.

```
4.602.3.4 template<typename _RandomNumberEngine, size_t __w, typename _UIntType > std::independent_bits_engine<
    _RandomNumberEngine, __w, _UIntType >::independent_bits_engine ( result_type __s ) [inline],
    [explicit]
```

Seed constructs a independent\_bits\_engine engine.

Constructs the underlying generator engine seeded with `__s`.

Parameters

<code>__s</code>	A seed value for the base class engine.
------------------	---

Definition at line 1120 of file random.h.

```
4.602.3.5 template<typename _RandomNumberEngine, size_t __w, typename _UIntType > template<typename _Sseq, typename
    = typename std::enable_if<!std::is_same<_Sseq, independent_bits_engine>::value && !std::is_same<_Sseq,
    _RandomNumberEngine>::value> ::type> std::independent_bits_engine< _RandomNumberEngine, __w,
    _UIntType >::independent_bits_engine ( _Sseq & __q ) [inline], [explicit]
```

Generator construct a independent\_bits\_engine engine.

Parameters

<code>__q</code>	A seed sequence.
------------------	------------------

Definition at line 1133 of file random.h.

#### 4.602.4 Member Function Documentation

4.602.4.1 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> const _RandomNumberEngine& std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::base( ) const [inline], [noexcept]`

Gets a const reference to the underlying generator engine object.

Definition at line 1168 of file random.h.

Referenced by `std::operator<<()`.

4.602.4.2 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> void std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::discard( unsigned long long __z ) [inline]`

Discard a sequence of random numbers.

Definition at line 1189 of file random.h.

4.602.4.3 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> static constexpr result_type std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::max( ) [inline], [static]`

Gets the maximum value in the generated random number range.

Definition at line 1182 of file random.h.

4.602.4.4 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> static constexpr result_type std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::min( ) [inline], [static]`

Gets the minimum value in the generated random number range.

Definition at line 1175 of file random.h.

4.602.4.5 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> result_type std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::operator()( )`

Gets the next value in the generated random number sequence.

4.602.4.6 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> void std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::seed( ) [inline]`

Reseeds the `independent_bits_engine` object with the default seed for the underlying base class generator engine.

Definition at line 1142 of file random.h.

4.602.4.7 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> void std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::seed( result_type __s ) [inline]`

Reseeds the `independent_bits_engine` object with the default seed for the underlying base class generator engine.

Definition at line 1150 of file random.h.

4.602.4.8 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> template<typename _Sseq> void std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::seed( _Sseq & __q ) [inline]`

Reseeds the `independent_bits_engine` object with the given seed sequence.

## Parameters

<code>__q</code>	A seed generator function.
------------------	----------------------------

Definition at line 1160 of file random.h.

## 4.602.5 Friends And Related Function Documentation

4.602.5.1 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> bool operator==( const independent_bits_engine< _RandomNumberEngine, __w, _UIntType> & __lhs, const independent_bits_engine< _RandomNumberEngine, __w, _UIntType> & __rhs ) [friend]`

Compares two independent\_bits\_engine random number generator objects of the same type for equality.

## Parameters

<code>__lhs</code>	A independent_bits_engine random number generator object.
<code>__rhs</code>	Another independent_bits_engine random number generator object.

## Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 1214 of file random.h.

4.602.5.2 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> template<typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& operator>>( std::basic_istream< _CharT, _Traits> & __is, std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType> & __x ) [friend]`

Extracts the current state of a % subtract\_with\_carry\_engine random number generator engine `__x` from the input stream `__is`.

## Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A independent_bits_engine random number generator engine.

## Returns

The input stream with the state of `__x` extracted or in an error state.

Definition at line 1232 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

## 4.603 std::indirect\_array&lt;\_Tp&gt; Class Template Reference

## Public Types

- `typedef _Tp value_type`

## Public Member Functions

- `indirect_array` (const `indirect_array` &)

- void **operator%=(const valarray<\_Tp> &) const**
- template<class \_Dom> void **operator%=(const \_Expr<\_Dom, \_Tp> &) const**
- void **operator&=(const valarray<\_Tp> &) const**
- template<class \_Dom> void **operator&=(const \_Expr<\_Dom, \_Tp> &) const**
- void **operator\*=(const valarray<\_Tp> &) const**
- template<class \_Dom> void **operator\*=(const \_Expr<\_Dom, \_Tp> &) const**
- void **operator+=(const valarray<\_Tp> &) const**
- template<class \_Dom> void **operator+=(const \_Expr<\_Dom, \_Tp> &) const**
- void **operator-=(const valarray<\_Tp> &) const**
- template<class \_Dom> void **operator-=(const \_Expr<\_Dom, \_Tp> &) const**
- void **operator/=(const valarray<\_Tp> &) const**
- template<class \_Dom> void **operator/=(const \_Expr<\_Dom, \_Tp> &) const**
- void **operator<=<=**(const valarray<\_Tp> &) const
- template<class \_Dom> void **operator<=<=**(const \_Expr<\_Dom, \_Tp> &) const
- **indirect\_array** & **operator=(const indirect\_array &)**
- void **operator=(const valarray<\_Tp> &) const**
- void **operator=(const \_Tp &) const**
- template<class \_Dom> void **operator=(const \_Expr<\_Dom, \_Tp> &) const**
- void **operator>=>=**(const valarray<\_Tp> &) const
- template<class \_Dom> void **operator>=>=**(const \_Expr<\_Dom, \_Tp> &) const
- void **operator^=(const valarray<\_Tp> &) const**
- template<class \_Dom> void **operator^=(const \_Expr<\_Dom, \_Tp> &) const**
- void **operator|=**(const valarray<\_Tp> &) const
- template<class \_Dom> void **operator|=**(const \_Expr<\_Dom, \_Tp> &) const

## Friends

- class **gslice\_array<\_Tp>**
- class **valarray<\_Tp>**

## 4.603.1 Detailed Description

**template<class \_Tp>class std::indirect\_array<\_Tp>**

Reference to arbitrary subset of an array.

An `indirect_array` is a reference to the actual elements of an array specified by an ordered array of indices. The way to get an `indirect_array` is to call `operator[]`(`valarray<size_t>`) on a `valarray`. The returned `indirect_array` then permits carrying operations out on the referenced subset of elements in the original `valarray`.

For example, if an `indirect_array` is obtained using the array (4,2,0) as an argument, and then assigned to an array containing (1,2,3), then the underlying array will have `array[0]==3`, `array[2]==2`, and `array[4]==1`.

## Parameters

<i>Tp</i>	Element type.
-----------	---------------

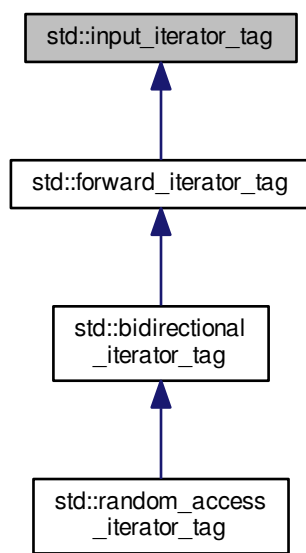
Definition at line 62 of file `indirect_array.h`.

The documentation for this class was generated from the following file:

- [indirect\\_array.h](#)

## 4.604 `std::input_iterator_tag` Struct Reference

Inheritance diagram for `std::input_iterator_tag`:



### 4.604.1 Detailed Description

Marking input iterators.

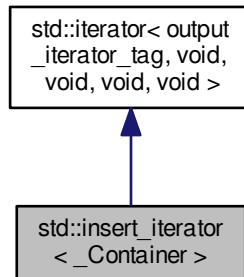
Definition at line 89 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

#### 4.605 `std::insert_iterator<_Container>` Class Template Reference

Inheritance diagram for `std::insert_iterator<_Container>`:



##### Public Types

- typedef `_Container` `container_type`
- typedef void `difference_type`
- typedef `output_iterator_tag` `iterator_category`
- typedef void `pointer`
- typedef void `reference`
- typedef void `value_type`

##### Public Member Functions

- `insert_iterator` (`_Container &__x`, `typename _Container::iterator __i`)
- `insert_iterator & operator*` ()
- `insert_iterator & operator++` ()
- `insert_iterator & operator++` (int)
- `insert_iterator & operator=` (const `typename _Container::value_type &__value`)
- `insert_iterator & operator=` (`typename _Container::value_type &&__value`)

##### Protected Attributes

- `_Container * container`
- `_Container::iterator iter`

##### 4.605.1 Detailed Description

```
template<typename _Container>class std::insert_iterator<_Container>
```

Turns assignment into insertion.

These are output iterators, constructed from a container-of-T. Assigning a T to the iterator inserts it in the container at the iterator's position, rather than overwriting the value at that position.

(Sequences will actually insert a *copy* of the value before the iterator's position.)

Tip: Using the inserter function to create these iterators can save typing.

Definition at line 587 of file stl\_iterator.h.

#### 4.605.2 Member Typedef Documentation

##### 4.605.2.1 template<typename \_Container> typedef \_Container std::insert\_iterator< \_Container >::container\_type

A nested typedef for the type of whatever container you used.

Definition at line 596 of file stl\_iterator.h.

##### 4.605.2.2 typedef void std::iterator< output\_iterator\_tag , void , void , void , void >::difference\_type [inherited]

Distance between iterators is represented as this type.

Definition at line 125 of file stl\_iterator\_base\_types.h.

##### 4.605.2.3 typedef output\_iterator\_tag std::iterator< output\_iterator\_tag , void , void , void , void >::iterator\_category [inherited]

One of the [tag types](#).

Definition at line 121 of file stl\_iterator\_base\_types.h.

##### 4.605.2.4 typedef void std::iterator< output\_iterator\_tag , void , void , void , void >::pointer [inherited]

This type represents a pointer-to-value\_type.

Definition at line 127 of file stl\_iterator\_base\_types.h.

##### 4.605.2.5 typedef void std::iterator< output\_iterator\_tag , void , void , void , void >::reference [inherited]

This type represents a reference-to-value\_type.

Definition at line 129 of file stl\_iterator\_base\_types.h.

##### 4.605.2.6 typedef void std::iterator< output\_iterator\_tag , void , void , void , void >::value\_type [inherited]

The type "pointed to" by the iterator.

Definition at line 123 of file stl\_iterator\_base\_types.h.

#### 4.605.3 Constructor & Destructor Documentation

##### 4.605.3.1 template<typename \_Container> std::insert\_iterator< \_Container >::insert\_iterator ( \_Container & \_\_x, typename \_Container::iterator \_\_i ) [inline]

The only way to create this iterator is with a container and an initial position (a normal iterator into the container).

Definition at line 602 of file stl\_iterator.h.

#### 4.605.4 Member Function Documentation



4.605.4.1 `template<typename _Container> insert_iterator& std::insert_iterator<_Container>::operator* ( )`  
`[inline]`

Simply returns `*this`.

Definition at line 656 of file `stl_iterator.h`.

4.605.4.2 `template<typename _Container> insert_iterator& std::insert_iterator<_Container>::operator++ ( )`  
`[inline]`

Simply returns `*this`. (This iterator does not *move*.)

Definition at line 661 of file `stl_iterator.h`.

4.605.4.3 `template<typename _Container> insert_iterator& std::insert_iterator<_Container>::operator++ ( int )`  
`[inline]`

Simply returns `*this`. (This iterator does not *move*.)

Definition at line 666 of file `stl_iterator.h`.

4.605.4.4 `template<typename _Container> insert_iterator& std::insert_iterator<_Container>::operator= ( const`  
`typename _Container::value_type & __value ) [inline]`

#### Parameters

<code>__value</code>	An instance of whatever type <code>container_type::const_reference</code> is; presumably a reference-to- <code>const T</code> for <code>container&lt;T&gt;</code> .
----------------------	---

#### Returns

This iterator, for chained operations.

This kind of iterator maintains its own position in the container. Assigning a value to the iterator will insert the value into the container at the place before the iterator.

The position is maintained such that subsequent assignments will insert values immediately after one another. For example,

```
// vector v contains A and Z
insert_iterator i (v, ++v.begin());
i = 1;
i = 2;
i = 3;

// vector v contains A, 1, 2, 3, and Z
```

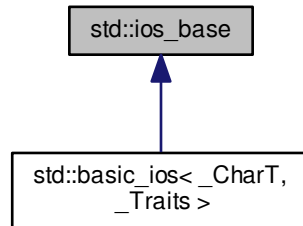
Definition at line 638 of file `stl_iterator.h`.

The documentation for this class was generated from the following file:

- [stl\\_iterator.h](#)

## 4.606 std::ios\_base Class Reference

Inheritance diagram for std::ios\_base:



## Classes

- class [failure](#)

## Public Types

- enum [event](#) { [erase\\_event](#), [imbue\\_event](#), [copyfmt\\_event](#) }
- typedef void(\* [event\\_callback](#)) ([event](#) \_\_e, [ios\\_base](#) &\_\_b, int \_\_i)
- typedef [\\_ios\\_Fmtflags](#) [fmtflags](#)
- typedef int [io\\_state](#)
- typedef [\\_ios\\_istate](#) [iostate](#)
- typedef int [open\\_mode](#)
- typedef [\\_ios\\_Openmode](#) [openmode](#)
- typedef int [seek\\_dir](#)
- typedef [\\_ios\\_Seekdir](#) [seekdir](#)
- typedef [std::streamoff](#) [streamoff](#)
- typedef [std::streampos](#) [streampos](#)

## Public Member Functions

- virtual [~ios\\_base](#) ()
- const [locale](#) & [\\_M\\_getloc](#) () const
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [locale](#) [getloc](#) () const
- [locale](#) [imbue](#) (const [locale](#) &\_\_loc) throw ()
- long & [iword](#) (int \_\_ix)
- [streamsize](#) [precision](#) () const
- [streamsize](#) [precision](#) ([streamsize](#) \_\_prec)
- void \*& [pword](#) (int \_\_ix)
- void [register\\_callback](#) ([event\\_callback](#) \_\_fn, int \_\_index)

- [fmtflags setf](#) ([fmtflags](#) \_\_fmtfl)
- [fmtflags setf](#) ([fmtflags](#) \_\_fmtfl, [fmtflags](#) \_\_mask)
- void [unsetf](#) ([fmtflags](#) \_\_mask)
- [streamsize width](#) () const
- [streamsize width](#) ([streamsize](#) \_\_wide)

#### Static Public Member Functions

- static bool [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static int [xalloc](#) () throw ()

#### Static Public Attributes

- static const [fmtflags adjustfield](#)
- static const [openmode app](#)
- static const [openmode ate](#)
- static const [iostate badbit](#)
- static const [fmtflags basefield](#)
- static const [seekdir beg](#)
- static const [openmode binary](#)
- static const [fmtflags boolalpha](#)
- static const [seekdir cur](#)
- static const [fmtflags dec](#)
- static const [seekdir end](#)
- static const [iostate eofbit](#)
- static const [iostate failbit](#)
- static const [fmtflags fixed](#)
- static const [fmtflags floatfield](#)
- static const [iostate goodbit](#)
- static const [fmtflags hex](#)
- static const [openmode in](#)
- static const [fmtflags internal](#)
- static const [fmtflags left](#)
- static const [fmtflags oct](#)
- static const [openmode out](#)
- static const [fmtflags right](#)
- static const [fmtflags scientific](#)
- static const [fmtflags showbase](#)
- static const [fmtflags showpoint](#)
- static const [fmtflags showpos](#)
- static const [fmtflags skipws](#)
- static const [openmode trunc](#)
- static const [fmtflags unitbuf](#)
- static const [fmtflags uppercase](#)

#### Protected Types

- enum { [\\_S\\_local\\_word\\_size](#) }

## Protected Member Functions

- void **\_M\_call\_callbacks** (event \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- \_Words & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()

## Protected Attributes

- \_Callback\_list \* **\_M\_callbacks**
- iostate **\_M\_exception**
- fmtflags **\_M\_flags**
- locale **\_M\_ios\_locale**
- \_Words **\_M\_local\_word** [\_S\_local\_word\_size]
- streamsize **\_M\_precision**
- iostate **\_M\_streambuf\_state**
- streamsize **\_M\_width**
- \_Words \* **\_M\_word**
- int **\_M\_word\_size**
- \_Words **\_M\_word\_zero**

## 4.606.1 Detailed Description

The base of the I/O class hierarchy.

This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see `ios_base` when they need to specify the full name of the various I/O flags (e.g., the openmodes).

Definition at line 199 of file `ios_base.h`.

## 4.606.2 Member Typedef Documentation

## 4.606.2.1 typedef void(\* std::ios\_base::event\_callback) (event \_\_e, ios\_base &amp;\_\_b, int \_\_i)

The type of an event callback function.

## Parameters

<code>__e</code>	One of the members of the event enum.
<code>__b</code>	Reference to the <code>ios_base</code> object.
<code>__i</code>	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 436 of file `ios_base.h`.

## 4.606.2.2 typedef \_Ios\_Fmtflags std::ios\_base::fmtflags

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 255 of file ios\_base.h.

#### 4.606.2.3 `typedef _Ios_Iostate std::ios_base::iostate`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 330 of file ios\_base.h.

#### 4.606.2.4 typedef \_Ios\_Openmode std::ios\_base::openmode

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 361 of file `ios_base.h`.

#### 4.606.2.5 typedef \_Ios\_Seekdir std::ios\_base::seekdir

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 393 of file `ios_base.h`.

### 4.606.3 Member Enumeration Documentation

#### 4.606.3.1 enum std::ios\_base::event

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 419 of file `ios_base.h`.

### 4.606.4 Constructor & Destructor Documentation

#### 4.606.4.1 virtual std::ios\_base::~ios\_base ( ) [virtual]

Invokes each callback with `erase_event`. Destroys local storage.

Note that the `ios_base` object for the standard streams never gets destroyed. As a result, any callbacks registered with the standard streams will not get invoked with `erase_event` (unless `copyfmt` is used).

#### 4.606.5 Member Function Documentation

##### 4.606.5.1 `const locale& std::ios_base::_M_getloc ( ) const` `[inline]`

Locale access.

##### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 706 of file `ios_base.h`.

##### 4.606.5.2 `fmtflags std::ios_base::flags ( ) const` `[inline]`

Access to format flags.

##### Returns

The format control flags for both input and output.

Definition at line 551 of file `ios_base.h`.

##### 4.606.5.3 `fmtflags std::ios_base::flags ( fmtflags __fmtfl )` `[inline]`

Setting new format flags all at once.

##### Parameters

<code>__fmtfl</code>	The new flags to set.
----------------------	-----------------------

##### Returns

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 562 of file `ios_base.h`.

##### 4.606.5.4 `locale std::ios_base::getloc ( ) const` `[inline]`

Locale access.

##### Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::ios_base::getloc()`, the global C++ locale.

Definition at line 695 of file `ios_base.h`.

##### 4.606.5.5 `locale std::ios_base::imbue ( const locale & __loc ) throw`

Setting a new locale.

## Parameters

<code>__loc</code>	The new locale.
--------------------	-----------------

## Returns

The previous locale.

Sets the new locale for this stream, and then invokes each callback with imbue\_event.

**4.606.5.6** `long& std::ios_base::iword ( int __ix ) [inline]`

Access to integer array.

## Parameters

<code>__ix</code>	Index into the array.
-------------------	-----------------------

## Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 741 of file ios\_base.h.

**4.606.5.7** `streamsize std::ios_base::precision ( ) const [inline]`

Flags access.

## Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 621 of file ios\_base.h.

**4.606.5.8** `streamsize std::ios_base::precision ( streamsize __prec ) [inline]`

Changing flags.

## Parameters

<code>__prec</code>	The new precision value.
---------------------	--------------------------

## Returns

The previous value of precision().

Definition at line 630 of file ios\_base.h.

**4.606.5.9** `void*& std::ios_base::pword ( int __ix ) [inline]`

Access to void pointer array.



**Parameters**

<code>__ix</code>	Index into the array.
-------------------	-----------------------

**Returns**

A reference to a `void*` associated with the index.

The `pwd` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 762 of file `ios_base.h`.

**4.606.5.10** `void std::ios_base::register_callback ( event_callback __fn, int __index )`

Add the callback `__fn` with parameter `__index`.

**Parameters**

<code>__fn</code>	The function to add.
<code>__index</code>	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**4.606.5.11** `fmtflags std::ios_base::setf ( fmtflags __fmtfl ) [inline]`

Setting new format flags.

**Parameters**

<code>__fmtfl</code>	Additional flags to set.
----------------------	--------------------------

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 578 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

**4.606.5.12** `fmtflags std::ios_base::setf ( fmtflags __fmtfl, fmtflags __mask ) [inline]`

Setting new format flags.

**Parameters**

<code>__fmtfl</code>	Additional flags to set.
<code>__mask</code>	The flags mask for <code>__fmtfl</code> .

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 595 of file `ios_base.h`.

**4.606.5.13** `static bool std::ios_base::sync_with_stdio( bool __sync = true ) [static]`

Interaction with the standard C I/O objects.

**Parameters**

<code>__sync</code>	Whether to synchronize or not.
---------------------	--------------------------------

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt1.html>

**4.606.5.14** `void std::ios_base::unsetf( fmtflags __mask ) [inline]`

Clearing format flags.

**Parameters**

<code>__mask</code>	The flags to unset.
---------------------	---------------------

This function clears `__mask` in the format flags.

Definition at line 610 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

**4.606.5.15** `streamsize std::ios_base::width( ) const [inline]`

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 644 of file `ios_base.h`.

**4.606.5.16** `streamsize std::ios_base::width( streamsize __wide ) [inline]`

Changing flags.

**Parameters**

<code>__wide</code>	The new width value.
---------------------	----------------------

**Returns**

The previous value of `width()`.

Definition at line 653 of file `ios_base.h`.

**4.606.5.17** `static int std::ios_base::xalloc ( ) throw` `[static]`

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**4.606.6 Member Data Documentation**

**4.606.6.1** `const fmtflags std::ios_base::adjustfield` `[static]`

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 310 of file `ios_base.h`.

Referenced by `std::internal()`, `std::left()`, and `std::right()`.

**4.606.6.2** `const openmode std::ios_base::app` `[static]`

Seek to end before each write.

Definition at line 364 of file `ios_base.h`.

**4.606.6.3** `const openmode std::ios_base::ate` `[static]`

Open and seek to end immediately after opening.

Definition at line 367 of file `ios_base.h`.

**4.606.6.4** `const iostate std::ios_base::badbit` `[static]`

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 334 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::bad()`, and `std::basic_ios<_CharT, _Traits>::fail()`.

**4.606.6.5** `const fmtflags std::ios_base::basefield` `[static]`

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Definition at line 313 of file `ios_base.h`.

Referenced by `std::dec()`, `std::hex()`, and `std::oct()`.

#### 4.606.6.6 `const seekdir std::ios_base::beg` [static]

Request a seek relative to the beginning of the stream.

Definition at line 396 of file `ios_base.h`.

#### 4.606.6.7 `const openmode std::ios_base::binary` [static]

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 372 of file `ios_base.h`.

#### 4.606.6.8 `const fmtflags std::ios_base::boolalpha` [static]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 258 of file `ios_base.h`.

Referenced by `std::boolalpha()`, and `std::noboolalpha()`.

#### 4.606.6.9 `const seekdir std::ios_base::cur` [static]

Request a seek relative to the current position within the sequence.

Definition at line 399 of file `ios_base.h`.

#### 4.606.6.10 `const fmtflags std::ios_base::dec` [static]

Converts integer input or generates integer output in decimal base.

Definition at line 261 of file `ios_base.h`.

Referenced by `std::dec()`.

#### 4.606.6.11 `const seekdir std::ios_base::end` [static]

Request a seek relative to the current end of the sequence.

Definition at line 402 of file `ios_base.h`.

#### 4.606.6.12 `const iostate std::ios_base::eofbit` [static]

Indicates that an input operation reached the end of an input sequence.

Definition at line 337 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::eof()`.

#### 4.606.6.13 `const iostate std::ios_base::failbit` [static]

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 342 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::fail()`.

#### 4.606.6.14 `const fmtflags std::ios_base::fixed` [static]

Generate floating-point output in fixed-point notation.

Definition at line 264 of file `ios_base.h`.

Referenced by `std::fixed()`.

**4.606.6.15** `const fmtflags std::ios_base::floatfield` `[static]`

A mask of `scientific|fixed`. Useful for the 2-arg form of `setf`.

Definition at line 316 of file `ios_base.h`.

Referenced by `std::fixed()`, and `std::scientific()`.

**4.606.6.16** `const iostate std::ios_base::goodbit` `[static]`

Indicates all is well.

Definition at line 345 of file `ios_base.h`.

**4.606.6.17** `const fmtflags std::ios_base::hex` `[static]`

Converts integer input or generates integer output in hexadecimal base.

Definition at line 267 of file `ios_base.h`.

Referenced by `std::hex()`.

**4.606.6.18** `const openmode std::ios_base::in` `[static]`

Open for input. Default for `ifstream` and `fstream`.

Definition at line 375 of file `ios_base.h`.

**4.606.6.19** `const fmtflags std::ios_base::internal` `[static]`

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 272 of file `ios_base.h`.

Referenced by `std::internal()`.

**4.606.6.20** `const fmtflags std::ios_base::left` `[static]`

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 276 of file `ios_base.h`.

Referenced by `std::left()`.

**4.606.6.21** `const fmtflags std::ios_base::oct` `[static]`

Converts integer input or generates integer output in octal base.

Definition at line 279 of file `ios_base.h`.

Referenced by `std::oct()`.

**4.606.6.22** `const openmode std::ios_base::out` `[static]`

Open for output. Default for `ofstream` and `fstream`.

Definition at line 378 of file `ios_base.h`.

**4.606.6.23** `const fmtflags std::ios_base::right` `[static]`

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 283 of file `ios_base.h`.

Referenced by `std::right()`.

**4.606.6.24** `const fmtflags std::ios_base::scientific` `[static]`

Generates floating-point output in scientific notation.

Definition at line 286 of file `ios_base.h`.

Referenced by `std::scientific()`.

**4.606.6.25** `const fmtflags std::ios_base::showbase` `[static]`

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 290 of file `ios_base.h`.

Referenced by `std::noshowbase()`, and `std::showbase()`.

**4.606.6.26** `const fmtflags std::ios_base::showpoint` `[static]`

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 294 of file `ios_base.h`.

Referenced by `std::noshowpoint()`, and `std::showpoint()`.

**4.606.6.27** `const fmtflags std::ios_base::showpos` `[static]`

Generates a + sign in non-negative generated numeric output.

Definition at line 297 of file `ios_base.h`.

Referenced by `std::noshowpos()`, and `std::showpos()`.

**4.606.6.28** `const fmtflags std::ios_base::skipws` `[static]`

Skips leading white space before certain input operations.

Definition at line 300 of file `ios_base.h`.

Referenced by `std::noskipws()`, and `std::skipws()`.

**4.606.6.29** `const openmode std::ios_base::trunc` `[static]`

Open for input. Default for `ofstream`.

Definition at line 381 of file `ios_base.h`.

**4.606.6.30** `const fmtflags std::ios_base::unitbuf` `[static]`

Flushes output after each output operation.

Definition at line 303 of file `ios_base.h`.

Referenced by `std::nounitbuf()`, and `std::unitbuf()`.

#### 4.606.6.31 `const fmtflags std::ios_base::uppercase` [static]

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 307 of file `ios_base.h`.

Referenced by `std::nouppercase()`, and `std::uppercase()`.

The documentation for this class was generated from the following file:

- [ios\\_base.h](#)

### 4.607 `std::ios_base::failure` Class Reference

Inherits exception.

#### Public Member Functions

- **failure** (const [string](#) &\_\_str) throw ()
- virtual const char \* **what** () const throw ()

#### 4.607.1 Detailed Description

These are thrown to indicate problems with io.

27.4.2.1.1 Class `ios_base::failure`.

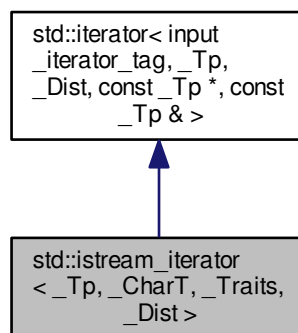
Definition at line 209 of file `ios_base.h`.

The documentation for this class was generated from the following file:

- [ios\\_base.h](#)

### 4.608 `std::istream_iterator< _Tp, _CharT, _Traits, _Dist >` Class Template Reference

Inheritance diagram for `std::istream_iterator< _Tp, _CharT, _Traits, _Dist >`:



## Public Types

- typedef `_CharT` **char\_type**
- typedef `_Dist` [difference\\_type](#)
- typedef `basic_istream< _CharT, _Traits >` **istream\_type**
- typedef [input\\_iterator\\_tag](#) **iterator\_category**
- typedef `const _Tp *` [pointer](#)
- typedef `const _Tp &` [reference](#)
- typedef `_Traits` **traits\_type**
- typedef `_Tp` [value\\_type](#)

## Public Member Functions

- constexpr [istream\\_iterator](#) ()
- [istream\\_iterator](#) ([istream\\_type](#) &\_\_s)
- **istream\_iterator** (const [istream\\_iterator](#) &\_\_obj)
- `bool` **\_M\_equal** (const [istream\\_iterator](#) &\_\_x) const
- `const _Tp &` **operator\*** () const
- [istream\\_iterator](#) & **operator++** ()
- [istream\\_iterator](#) **operator++** (int)
- `const _Tp *` **operator->** () const

## 4.608.1 Detailed Description

template<typename `_Tp`, typename `_CharT` = char, typename `_Traits` = `char_traits<_CharT>`, typename `_Dist` = `ptrdiff_t`> class std::istream\_iterator< `_Tp`, `_CharT`, `_Traits`, `_Dist` >

Provides input iterator semantics for streams.

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

## 4.608.2 Member Typedef Documentation

4.608.2.1 typedef `_Dist` std::iterator< [input\\_iterator\\_tag](#) , `_Tp`, `_Dist` , `const _Tp *` , `const _Tp &` >::[difference\\_type](#) [\[inherited\]](#)

Distance between iterators is represented as this type.

Definition at line 125 of file `stl_iterator_base_types.h`.

4.608.2.2 typedef [input\\_iterator\\_tag](#) std::iterator< [input\\_iterator\\_tag](#) , `_Tp`, `_Dist` , `const _Tp *` , `const _Tp &` >::[iterator\\_category](#) [\[inherited\]](#)

One of the [tag types](#).

Definition at line 121 of file `stl_iterator_base_types.h`.

4.608.2.3 typedef `const _Tp *` std::iterator< [input\\_iterator\\_tag](#) , `_Tp`, `_Dist` , `const _Tp *` , `const _Tp &` >::[pointer](#) [\[inherited\]](#)

This type represents a pointer-to-value\_type.

Definition at line 127 of file `stl_iterator_base_types.h`.



**4.608.2.4** `typedef const _Tp & std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp & >::reference`  
[inherited]

This type represents a reference-to-value\_type.

Definition at line 129 of file `stl_iterator_base_types.h`.

**4.608.2.5** `typedef _Tp std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp & >::value_type`  
[inherited]

The type "pointed to" by the iterator.

Definition at line 123 of file `stl_iterator_base_types.h`.

### 4.608.3 Constructor & Destructor Documentation

**4.608.3.1** `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t>`  
`constexpr std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator ( )` [inline]

Construct end of input stream iterator.

Definition at line 64 of file `stream_iterator.h`.

**4.608.3.2** `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t>`  
`std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator ( istream_type & __s )` [inline]

Construct start of input stream iterator.

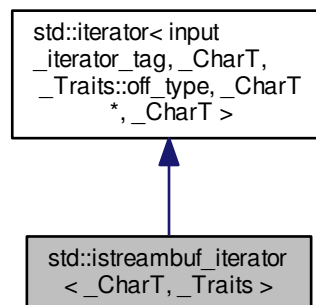
Definition at line 68 of file `stream_iterator.h`.

The documentation for this class was generated from the following file:

- [stream\\_iterator.h](#)

## 4.609 std::istreambuf\_iterator< \_CharT, \_Traits > Class Template Reference

Inheritance diagram for `std::istreambuf_iterator< _CharT, _Traits >`:



## Public Types

- typedef \_Traits::off\_type [difference\\_type](#)
- typedef [input\\_iterator\\_tag](#) iterator\_category
- typedef \_CharT \* [pointer](#)
- typedef \_CharT [reference](#)
- typedef \_CharT [value\\_type](#)
- typedef \_CharT [char\\_type](#)
- typedef \_Traits [traits\\_type](#)
- typedef \_Traits::int\_type [int\\_type](#)
- typedef basic\_streambuf<\_CharT, \_Traits> [streambuf\\_type](#)
- typedef basic\_istream<\_CharT, \_Traits> [istream\\_type](#)

## Public Member Functions

- constexpr [istreambuf\\_iterator](#) () noexcept
- **istreambuf\_iterator** (const [istreambuf\\_iterator](#) &) noexcept=default
- [istreambuf\\_iterator](#) ([istream\\_type](#) &\_\_s) noexcept
- [istreambuf\\_iterator](#) ([streambuf\\_type](#) \*\_\_s) noexcept
- bool [equal](#) (const [istreambuf\\_iterator](#) &\_\_b) const
- [char\\_type](#) operator\* () const
- [istreambuf\\_iterator](#) & operator++ ()
- [istreambuf\\_iterator](#) operator++ (int)

## Friends

- template<bool \_\_IsMove, typename \_CharT2> \_\_gnu\_cxx::\_\_enable\_if<\_\_is\_char<\_CharT2>::\_\_value, \_CharT2 \* >::\_\_type **copy\_move\_a2** ([istreambuf\\_iterator](#)<\_CharT2>, [istreambuf\\_iterator](#)<\_CharT2>, \_CharT2 \*)
- template<typename \_CharT2> \_\_gnu\_cxx::\_\_enable\_if<\_\_is\_char<\_CharT2>::\_\_value, [ostreambuf\\_iterator](#)<\_CharT2> >::\_\_type **copy** ([istreambuf\\_iterator](#)<\_CharT2>, [istreambuf\\_iterator](#)<\_CharT2>, [ostreambuf\\_iterator](#)<\_CharT2>)
- template<typename \_CharT2> \_\_gnu\_cxx::\_\_enable\_if<\_\_is\_char<\_CharT2>::\_\_value, [istreambuf\\_iterator](#)<\_CharT2> >::\_\_type **find** ([istreambuf\\_iterator](#)<\_CharT2>, [istreambuf\\_iterator](#)<\_CharT2>, const \_CharT2 &)

## 4.609.1 Detailed Description

template<typename \_CharT, typename \_Traits> class std::istreambuf\_iterator<\_CharT, \_Traits>

Provides input iterator semantics for streambufs.

Definition at line 399 of file stl\_algobase.h.

## 4.609.2 Member Typedef Documentation

4.609.2.1 template<typename \_CharT, typename \_Traits> typedef \_CharT std::istreambuf\_iterator<\_CharT, \_Traits>::\_\_char\_type

Public typedefs.

Definition at line 64 of file streambuf\_iterator.h.

**4.609.2.2** `typedef _Traits::off_type std::iterator< input_iterator_tag , _CharT , _Traits::off_type , _CharT * , _CharT >::difference_type` [inherited]

Distance between iterators is represented as this type.

Definition at line 125 of file `stl_iterator_base_types.h`.

**4.609.2.3** `template<typename _CharT , typename _Traits > typedef _Traits::int_type std::istreambuf_iterator< _CharT, _Traits >::int_type`

Public typedefs.

Definition at line 66 of file `streambuf_iterator.h`.

**4.609.2.4** `template<typename _CharT , typename _Traits > typedef basic_istream< _CharT, _Traits > std::istreambuf_iterator< _CharT, _Traits >::istream_type`

Public typedefs.

Definition at line 68 of file `streambuf_iterator.h`.

**4.609.2.5** `typedef input_iterator_tag std::iterator< input_iterator_tag , _CharT , _Traits::off_type , _CharT * , _CharT >::iterator_category` [inherited]

One of the [tag types](#).

Definition at line 121 of file `stl_iterator_base_types.h`.

**4.609.2.6** `typedef _CharT * std::iterator< input_iterator_tag , _CharT , _Traits::off_type , _CharT * , _CharT >::pointer` [inherited]

This type represents a pointer-to-value\_type.

Definition at line 127 of file `stl_iterator_base_types.h`.

**4.609.2.7** `typedef _CharT std::iterator< input_iterator_tag , _CharT , _Traits::off_type , _CharT * , _CharT >::reference` [inherited]

This type represents a reference-to-value\_type.

Definition at line 129 of file `stl_iterator_base_types.h`.

**4.609.2.8** `template<typename _CharT , typename _Traits > typedef basic_streambuf< _CharT, _Traits > std::istreambuf_iterator< _CharT, _Traits >::streambuf_type`

Public typedefs.

Definition at line 67 of file `streambuf_iterator.h`.

**4.609.2.9** `template<typename _CharT , typename _Traits > typedef _Traits std::istreambuf_iterator< _CharT, _Traits >::traits_type`

Public typedefs.

Definition at line 65 of file `streambuf_iterator.h`.

**4.609.2.10** `typedef _CharT std::iterator< input_iterator_tag , _CharT , _Traits::off_type , _CharT * , _CharT >::value_type` [inherited]

The type "pointed to" by the iterator.

Definition at line 123 of file stl\_iterator\_base\_types.h.

#### 4.609.3 Constructor & Destructor Documentation

4.609.3.1 `template<typename _CharT, typename _Traits> constexpr std::istreambuf_iterator< _CharT, _Traits>::istreambuf_iterator( ) [inline], [noexcept]`

Construct end of input stream iterator.

Definition at line 102 of file streambuf\_iterator.h.

4.609.3.2 `template<typename _CharT, typename _Traits> std::istreambuf_iterator< _CharT, _Traits>::istreambuf_iterator( istream_type &__s ) [inline], [noexcept]`

Construct start of input stream iterator.

Definition at line 112 of file streambuf\_iterator.h.

4.609.3.3 `template<typename _CharT, typename _Traits> std::istreambuf_iterator< _CharT, _Traits>::istreambuf_iterator( streambuf_type *__s ) [inline], [noexcept]`

Construct start of streambuf iterator.

Definition at line 116 of file streambuf\_iterator.h.

#### 4.609.4 Member Function Documentation

4.609.4.1 `template<typename _CharT, typename _Traits> bool std::istreambuf_iterator< _CharT, _Traits>::equal( const istreambuf_iterator< _CharT, _Traits> &__b ) const [inline]`

Return true both iterators are end or both are not end.

Definition at line 172 of file streambuf\_iterator.h.

4.609.4.2 `template<typename _CharT, typename _Traits> char_type std::istreambuf_iterator< _CharT, _Traits>::operator*( ) const [inline]`

Return the current character pointed to by iterator. This returns streambuf.sgetc(). It cannot be assigned. NB: The result of operator\*() on an end of stream is undefined.

Definition at line 123 of file streambuf\_iterator.h.

4.609.4.3 `template<typename _CharT, typename _Traits> istreambuf_iterator& std::istreambuf_iterator< _CharT, _Traits>::operator++( ) [inline]`

Advance the iterator. Calls streambuf.sbumpc().

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

4.609.4.4 `template<typename _CharT, typename _Traits> istreambuf_iterator std::istreambuf_iterator< _CharT, _Traits>::operator++( int ) [inline]`

Advance the iterator. Calls streambuf.sbumpc().

Definition at line 152 of file streambuf\_iterator.h.

The documentation for this class was generated from the following files:

- [stl\\_algobase.h](#)

- [streambuf\\_iterator.h](#)

#### 4.610 `std::iterator<_Category, _Tp, _Distance, _Pointer, _Reference>` Struct Template Reference

##### Public Types

- typedef `_Distance` [difference\\_type](#)
- typedef `_Category` [iterator\\_category](#)
- typedef `_Pointer` [pointer](#)
- typedef `_Reference` [reference](#)
- typedef `_Tp` [value\\_type](#)

##### 4.610.1 Detailed Description

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference =
_Tp> struct std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >
```

Common iterator class.

This class does nothing but define nested typedefs. Iterator classes can inherit from this class to save some work. The typedefs are then used in specializations and overloading.

In particular, there are no default implementations of requirements such as `operator++` and the like. (How could there be?)

Definition at line 118 of file `stl_iterator_base_types.h`.

##### 4.610.2 Member Typedef Documentation

4.610.2.1 `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*,  
typename _Reference = _Tp&> typedef _Distance std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference  
>::difference_type`

Distance between iterators is represented as this type.

Definition at line 125 of file `stl_iterator_base_types.h`.

4.610.2.2 `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*,  
typename _Reference = _Tp&> typedef _Category std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference  
>::iterator_category`

One of the [tag types](#).

Definition at line 121 of file `stl_iterator_base_types.h`.

4.610.2.3 `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename  
_Reference = _Tp&> typedef _Pointer std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::pointer`

This type represents a pointer-to-value\_type.

Definition at line 127 of file `stl_iterator_base_types.h`.

4.610.2.4 `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename  
_Reference = _Tp&> typedef _Reference std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::reference`

This type represents a reference-to-value\_type.

Definition at line 129 of file `stl_iterator_base_types.h`.

4.610.2.5 `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&> typedef _Tp std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::value_type`

The type "pointed to" by the iterator.

Definition at line 123 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 4.611 `std::iterator_traits< _Tp * >` Struct Template Reference

### Public Types

- typedef ptrdiff\_t **difference\_type**
- typedef [random\\_access\\_iterator\\_tag](#) **iterator\_category**
- typedef \_Tp \* **pointer**
- typedef \_Tp & **reference**
- typedef \_Tp **value\_type**

### 4.611.1 Detailed Description

`template<typename _Tp>struct std::iterator_traits< _Tp * >`

Partial specialization for pointer types.

Definition at line 175 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 4.612 `std::iterator_traits< const _Tp * >` Struct Template Reference

### Public Types

- typedef ptrdiff\_t **difference\_type**
- typedef [random\\_access\\_iterator\\_tag](#) **iterator\_category**
- typedef const \_Tp \* **pointer**
- typedef const \_Tp & **reference**
- typedef \_Tp **value\_type**

### 4.612.1 Detailed Description

`template<typename _Tp>struct std::iterator_traits< const _Tp * >`

Partial specialization for const pointer types.

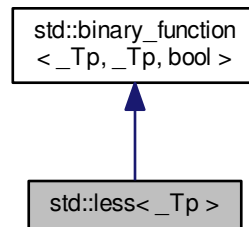
Definition at line 186 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

#### 4.613 std::less< \_Tp > Struct Template Reference

Inheritance diagram for std::less< \_Tp >:



##### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

##### Public Member Functions

- `bool` **operator()** (`const _Tp &__x`, `const _Tp &__y`) `const`

##### 4.613.1 Detailed Description

`template<typename _Tp>struct std::less< _Tp >`

One of the [comparison functors](#).

Definition at line 231 of file `stl_function.h`.

##### 4.613.2 Member Typedef Documentation

**4.613.2.1** typedef `_Tp` `std::binary_function< _Tp, _Tp, bool >::first_argument_type` `[inherited]`

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

**4.613.2.2** typedef `bool` `std::binary_function< _Tp, _Tp, bool >::result_type` `[inherited]`

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.613.2.3 `typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

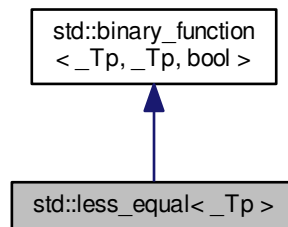
Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.614 std::less\_equal< \_Tp > Struct Template Reference

Inheritance diagram for `std::less_equal< _Tp >`:



### Public Types

- `typedef _Tp` [first\\_argument\\_type](#)
- `typedef bool` [result\\_type](#)
- `typedef _Tp` [second\\_argument\\_type](#)

### Public Member Functions

- `bool` **operator()** (const `_Tp` &\_\_x, const `_Tp` &\_\_y) const

### 4.614.1 Detailed Description

`template<typename _Tp>struct std::less_equal< _Tp >`

One of the [comparison functors](#).

Definition at line 249 of file `stl_function.h`.

### 4.614.2 Member Typedef Documentation

4.614.2.1 `typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument



Definition at line 117 of file `stl_function.h`.

**4.614.2.2** `typedef bool std::binary_function<_Tp, _Tp, bool>::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

**4.614.2.3** `typedef _Tp std::binary_function<_Tp, _Tp, bool>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.615 `std::linear_congruential_engine<_UIntType, __a, __c, __m>` Class Template Reference

### Public Types

- `typedef _UIntType result_type`

### Public Member Functions

- `linear_congruential_engine` (`result_type __s=default_seed`)
- `template<typename _Sseq, typename = typename std::enable_if<!std::is_same<_Sseq, linear_congruential_engine>::value>::type> linear_congruential_engine` (`_Sseq &__q`)
- `void discard` (`unsigned long long __z`)
- `result_type operator()` ()
- `void seed` (`result_type __s=default_seed`)
- `template<typename _Sseq> std::enable_if< std::is_class<_Sseq>::value>::type seed` (`_Sseq &__q`)

### Static Public Member Functions

- `static constexpr result_type max` ()
- `static constexpr result_type min` ()

### Static Public Attributes

- `static constexpr result_type default_seed`
- `static constexpr result_type increment`
- `static constexpr result_type modulus`
- `static constexpr result_type multiplier`

### Friends

- `template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT, typename _Traits> std::basic_ostream<_CharT, _Traits> & operator<<` (`std::basic_ostream<_CharT, _Traits> &__os, const std::linear_congruential_engine<_UIntType1, __a1, __c1, __m1> &__lcr`)
- `bool operator==` (`const linear_congruential_engine &__lhs, const linear_congruential_engine &__rhs`)

- `template<typename _UIntType1 , _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT , typename _Traits > std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, std::linear_congruential_engine< _UIntType1, __a1, __c1, __m1 > & __lcr)`

#### 4.615.1 Detailed Description

`template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> class std::linear_congruential_engine< _UIntType, __a, __c, __m >`

A model of a linear congruential random number generator.

A random number generator that produces pseudorandom numbers via linear function:

$$x_{i+1} \leftarrow (ax_i + c) \bmod m$$

The template parameter `_UIntType` must be an unsigned integral type large enough to store values up to `(__m-1)`. If the template parameter `__m` is 0, the modulus `__m` used is `std::numeric_limits<_UIntType>::max()` plus 1. Otherwise, the template parameters `__a` and `__c` must be less than `__m`.

The size of the state is 1.

Definition at line 241 of file `random.h`.

#### 4.615.2 Member Typedef Documentation

4.615.2.1 `template<typename _UIntType , _UIntType __a, _UIntType __c, _UIntType __m> typedef _UIntType std::linear_congruential_engine< _UIntType, __a, __c, __m >::result_type`

The type of the generated random value.

Definition at line 244 of file `random.h`.

#### 4.615.3 Constructor & Destructor Documentation

4.615.3.1 `template<typename _UIntType , _UIntType __a, _UIntType __c, _UIntType __m> std::linear_congruential_engine< _UIntType, __a, __c, __m >::linear_congruential_engine( result_type __s = default_seed )`  
`[inline], [explicit]`

Constructs a `linear_congruential_engine` random number generator engine with seed `__s`. The default seed value is 1.

Parameters

<code>__s</code>	The initial seed value.
------------------	-------------------------

Definition at line 268 of file `random.h`.

References `std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed()`.

4.615.3.2 `template<typename _UIntType , _UIntType __a, _UIntType __c, _UIntType __m> template<typename _Sseq , typename = typename std::enable_if<!std::is_same<_Sseq, linear_congruential_engine>::value>::type> std::linear_congruential_engine< _UIntType, __a, __c, __m >::linear_congruential_engine( _Sseq & __q )`  
`[inline], [explicit]`

Constructs a `linear_congruential_engine` random number generator engine seeded from the seed sequence `__q`.

## Parameters

<code>__q</code>	the seed sequence.
------------------	--------------------

Definition at line 281 of file random.h.

References `std::linear_congruential_engine<_UIntType, __a, __c, __m>::seed()`.

## 4.615.4 Member Function Documentation

4.615.4.1 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> void std::linear_congruential_engine<_UIntType, __a, __c, __m>::discard ( unsigned long long __z )`  
[inline]

Discard a sequence of random numbers.

Definition at line 325 of file random.h.

4.615.4.2 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> static constexpr result_type std::linear_congruential_engine<_UIntType, __a, __c, __m>::max ( )` [inline], [static]

Gets the largest possible value in the output range.

Definition at line 318 of file random.h.

4.615.4.3 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> static constexpr result_type std::linear_congruential_engine<_UIntType, __a, __c, __m>::min ( )` [inline], [static]

Gets the smallest possible value in the output range.

The minimum depends on the `__c` parameter: if it is zero, the minimum generated must be  $> 0$ , otherwise 0 is allowed.

Definition at line 311 of file random.h.

4.615.4.4 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> result_type std::linear_congruential_engine<_UIntType, __a, __c, __m>::operator() ( )` [inline]

Gets the next random number in the sequence.

Definition at line 335 of file random.h.

4.615.4.5 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> void std::linear_congruential_engine<_UIntType, __a, __c, __m>::seed ( result_type __s = default_seed )`

Reseeds the `linear_congruential_engine` random number generator engine sequence to the seed `__s`.

## Parameters

<code>__s</code>	The new seed.
------------------	---------------

Referenced by `std::linear_congruential_engine<_UIntType, __a, __c, __m>::linear_congruential_engine()`.

4.615.4.6 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> template<typename _Sseq> std::enable_if<std::is_class<_Sseq>::value>::type std::linear_congruential_engine<_UIntType, __a, __c, __m>::seed ( _Sseq & __q )`

Reseeds the `linear_congruential_engine` random number generator engine sequence using values from the seed sequence `__q`.

## Parameters

<code>__q</code>	the seed sequence.
------------------	--------------------

## 4.615.5 Friends And Related Function Documentation

4.615.5.1 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT, typename _Traits> std::basic_ostream<_CharT, _Traits>& operator<< ( std::basic_ostream<_CharT, _Traits> & __os, const std::linear_congruential_engine<_UIntType1, __a1, __c1, __m1> & __lcr ) [friend]`

Writes the textual representation of the state  $x(i)$  of  $x$  to `__os`.

## Parameters

<code>__os</code>	The output stream.
<code>__lcr</code>	A % linear_congruential_engine random number generator.

## Returns

`__os`.

4.615.5.2 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> bool operator== ( const linear_congruential_engine<_UIntType, __a, __c, __m> & __lhs, const linear_congruential_engine<_UIntType, __a, __c, __m> & __rhs ) [friend]`

Compares two linear congruential random number generator objects of the same type for equality.

## Parameters

<code>__lhs</code>	A linear congruential random number generator object.
<code>__rhs</code>	Another linear congruential random number generator object.

## Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 353 of file random.h.

4.615.5.3 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<_CharT, _Traits> & __is, std::linear_congruential_engine<_UIntType1, __a1, __c1, __m1> & __lcr ) [friend]`

Sets the state of the engine by reading its textual representation from `__is`.

The textual representation must have been previously written using an output stream whose imbued locale and whose type's template specialization arguments `_CharT` and `_Traits` were the same as those of `__is`.

## Parameters

<code>__is</code>	The input stream.
-------------------	-------------------

<code>__lcr</code>	A % linear_congruential_engine random number generator.
--------------------	---

#### Returns

`__is`.

#### 4.615.6 Member Data Documentation

4.615.6.1 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> constexpr result_type std::linear_congruential_engine<_UIntType, __a, __c, __m>::increment [static]`

An increment.

Definition at line 255 of file random.h.

4.615.6.2 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> constexpr result_type std::linear_congruential_engine<_UIntType, __a, __c, __m>::modulus [static]`

The modulus.

Definition at line 257 of file random.h.

4.615.6.3 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> constexpr result_type std::linear_congruential_engine<_UIntType, __a, __c, __m>::multiplier [static]`

The multiplier.

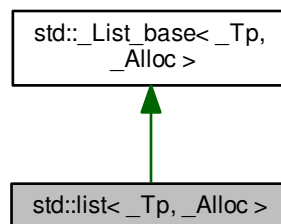
Definition at line 253 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

#### 4.616 `std::list<_Tp, _Alloc>` Class Template Reference

Inheritance diagram for `std::list<_Tp, _Alloc>`:



## Public Types

- typedef `_Alloc allocator_type`
- typedef `_List_const_iterator<_Tp> const_iterator`
- typedef `_Tp_alloc_type::const_pointer const_pointer`
- typedef `_Tp_alloc_type::const_reference const_reference`
- typedef `std::reverse_iterator<const_iterator> const_reverse_iterator`
- typedef `ptrdiff_t difference_type`
- typedef `_List_iterator<_Tp> iterator`
- typedef `_Tp_alloc_type::pointer pointer`
- typedef `_Tp_alloc_type::reference reference`
- typedef `std::reverse_iterator<iterator> reverse_iterator`
- typedef `size_t size_type`
- typedef `_Tp value_type`

## Public Member Functions

- `list()`
- `list(allocator_type &__a)`
- `list(size_type __n)`
- `list(size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())`
- `list(const list &__x)`
- `list(list &&__x) noexcept`
- `list(initializer_list<value_type> __l, const allocator_type &__a=allocator_type())`
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>> list(_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())`
- `void assign(size_type __n, const value_type &__val)`
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>> void assign(_InputIterator __first, _InputIterator __last)`
- `void assign(initializer_list<value_type> __l)`
- `reference back()`
- `const_reference back() const`
- `iterator begin() noexcept`
- `const_iterator begin() const noexcept`
- `const_iterator cbegin() const noexcept`
- `const_iterator cend() const noexcept`
- `void clear() noexcept`
- `const_reverse_iterator crbegin() const noexcept`
- `const_reverse_iterator crend() const noexcept`
- `template<typename... _Args> iterator emplace(iterator __position, _Args &&... __args)`
- `template<typename... _Args> void emplace_back(_Args &&... __args)`
- `template<typename... _Args> void emplace_front(_Args &&... __args)`
- `bool empty() const noexcept`
- `iterator end() noexcept`
- `const_iterator end() const noexcept`
- `iterator erase(iterator __position)`
- `iterator erase(iterator __first, iterator __last)`
- `reference front()`
- `const_reference front() const`
- `allocator_type get_allocator() const noexcept`
- `iterator insert(iterator __position, const value_type &__x)`

- `iterator insert (iterator __position, value_type &&__x)`
- `void insert (iterator __p, initializer_list< value_type > __l)`
- `void insert (iterator __position, size_type __n, const value_type &__x)`
- `template<typename _InputIterator, typename = std::::RequireInputIter<_InputIterator>> void insert (iterator __position, _InputIterator __first, _InputIterator __last)`
- `size_type max_size () const noexcept`
- `void merge (list &&__x)`
- `void merge (list &__x)`
- `template<typename _StrictWeakOrdering > void merge (list &&__x, _StrictWeakOrdering __comp)`
- `template<typename _StrictWeakOrdering > void merge (list &__x, _StrictWeakOrdering __comp)`
- `list & operator= (const list &__x)`
- `list & operator= (list &&__x)`
- `list & operator= (initializer_list< value_type > __l)`
- `void pop_back ()`
- `void pop_front ()`
- `void push_back (const value_type &__x)`
- `void push_back (value_type &&__x)`
- `void push_front (const value_type &__x)`
- `void push_front (value_type &&__x)`
- `reverse_iterator rbegin () noexcept`
- `const_reverse_iterator rend () const noexcept`
- `void remove (const _Tp &__value)`
- `template<typename _Predicate > void remove_if (_Predicate)`
- `reverse_iterator rend () noexcept`
- `const_reverse_iterator rend () const noexcept`
- `void resize (size_type __new_size)`
- `void resize (size_type __new_size, const value_type &__x)`
- `void reverse () noexcept`
- `size_type size () const noexcept`
- `void sort ()`
- `template<typename _StrictWeakOrdering > void sort (_StrictWeakOrdering)`
- `void splice (iterator __position, list &&__x)`
- `void splice (iterator __position, list &__x)`
- `void splice (iterator __position, list &&__x, iterator __i)`
- `void splice (iterator __position, list &__x, iterator __i)`
- `void splice (iterator __position, list &&__x, iterator __first, iterator __last)`
- `void splice (iterator __position, list &__x, iterator __first, iterator __last)`
- `void swap (list &__x)`
- `void unique ()`
- `template<typename _BinaryPredicate > void unique (_BinaryPredicate)`

## Protected Types

- `typedef _List_node< _Tp > _Node`

## Protected Member Functions

- `template<typename _Integer> void _M_assign_dispatch (_Integer __n, _Integer __val, __true_type)`
- `template<typename _InputIterator> void _M_assign_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `void _M_check_equal_allocators (list &__x)`
- `void _M_clear ()`
- `template<typename... _Args> _Node * _M_create_node (_Args &&... __args)`
- `void _M_default_append (size_type __n)`
- `void _M_default_initialize (size_type __n)`
- `void _M_erase (iterator __position)`
- `void _M_fill_assign (size_type __n, const value_type &__val)`
- `void _M_fill_initialize (size_type __n, const value_type &__x)`
- `_List_node<_Tp> * _M_get_node ()`
- `_Node_alloc_type & _M_get_Node_allocator () noexcept`
- `const _Node_alloc_type & _M_get_Node_allocator () const noexcept`
- `_Tp_alloc_type _M_get_Tp_allocator () const noexcept`
- `void _M_init ()`
- `template<typename _Integer> void _M_initialize_dispatch (_Integer __n, _Integer __x, __true_type)`
- `template<typename _InputIterator> void _M_initialize_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `template<typename... _Args> void _M_insert (iterator __position, _Args &&... __args)`
- `void _M_put_node (_List_node<_Tp> * __p)`
- `void _M_transfer (iterator __position, iterator __first, iterator __last)`

## Protected Attributes

- `_List_impl _M_impl`

## 4.616.1 Detailed Description

`template<typename _Tp, typename _Alloc = std::allocator<_Tp>> class std::list<_Tp, _Alloc>`

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

## Template Parameters

<code>_Tp</code>	Type of element.
<code>_Alloc</code>	Allocator type, defaults to <code>allocator&lt;_Tp&gt;</code> .

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#) with the exception of `at` and `operator[]`.

This is a *doubly linked* list. Traversal up and down the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike `std::vector` and `std::deque`, random-access iterators are not provided, so subscripting ( `[]` ) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.

Also unlike the other standard containers, `std::list` provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.

A couple points on memory allocation for `list<Tp>`:

First, we never actually allocate a `Tp`, we allocate `List_node<Tp>`'s and trust [20.1.5]/4 to DTRT. This is to ensure that after elements from `list<X, Alloc1>` are spliced into `list<X, Alloc2>`, destroying the memory of the second list is a valid operation, i.e., `Alloc1` giveth and `Alloc2` taketh away.



Second, a list conceptually represented as

A <----> B <----> C <----> D

is actually circular; a link exists between A and D. The list class holds (as its only data member) a private `list::iterator` pointing to *D*, not to *A*! To get to the head of the list, we start at the tail and move forward by one. When this member iterator's next/previous pointers refer to itself, the list is empty.

Definition at line 438 of file `stl_list.h`.

#### 4.616.2 Constructor & Destructor Documentation

**4.616.2.1** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list<_Tp, _Alloc>::list ( ) [inline]`

Default constructor creates no elements.

Definition at line 523 of file `stl_list.h`.

**4.616.2.2** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list<_Tp, _Alloc>::list ( const allocator_type &__a ) [inline], [explicit]`

Creates a list with no elements.

Parameters

<code>__a</code>	An allocator object.
------------------	----------------------

Definition at line 531 of file `stl_list.h`.

**4.616.2.3** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list<_Tp, _Alloc>::list ( size_type __n ) [inline], [explicit]`

Creates a list with default constructed elements.

Parameters

<code>__n</code>	The number of elements to initially create.
------------------	---

This constructor fills the list with `__n` default constructed elements.

Definition at line 543 of file `stl_list.h`.

**4.616.2.4** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list<_Tp, _Alloc>::list ( size_type __n, const value_type &__value, const allocator_type &__a = allocator_type() ) [inline]`

Creates a list with copies of an exemplar element.

Parameters

<code>__n</code>	The number of elements to initially create.
<code>__value</code>	An element to copy.
<code>__a</code>	An allocator object.

This constructor fills the list with `__n` copies of `__value`.

Definition at line 555 of file `stl_list.h`.

**4.616.2.5** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list<_Tp, _Alloc>::list ( const list<_Tp, _Alloc> &__x ) [inline]`

List copy constructor.

## Parameters

<code>__x</code>	A list of identical element and allocator types.
------------------	--

The newly-created list uses a copy of the allocation object used by `__x`.

Definition at line 582 of file `stl_list.h`.

**4.616.2.6** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list<_Tp, _Alloc>::list ( list<_Tp, _Alloc> && __x ) [inline], [noexcept]`

List move constructor.

## Parameters

<code>__x</code>	A list of identical element and allocator types.
------------------	--

The newly-created list contains the exact contents of `__x`. The contents of `__x` are a valid, but unspecified list.

Definition at line 594 of file `stl_list.h`.

**4.616.2.7** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list<_Tp, _Alloc>::list ( initializer_list<value_type> & __l, const allocator_type & __a = allocator_type() ) [inline]`

Builds a list from an `initializer_list`.

## Parameters

<code>__l</code>	An <code>initializer_list</code> of <code>value_type</code> .
<code>__a</code>	An allocator object.

Create a list consisting of copies of the elements in the `initializer_list` `__l`. This is linear in `__l.size()`.

Definition at line 605 of file `stl_list.h`.

**4.616.2.8** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator, typename = std::RequireInputIter<_InputIterator>> std::list<_Tp, _Alloc>::list ( _InputIterator __first, _InputIterator __last, const allocator_type & __a = allocator_type() ) [inline]`

Builds a list from a range.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__a</code>	An allocator object.

Create a list consisting of copies of the elements from `[__first, __last)`. This is linear in `N` (where `N` is `distance(__first, __last)`).

Definition at line 624 of file `stl_list.h`.

**4.616.3 Member Function Documentation**

**4.616.3.1** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename... _Args> _Node* std::list<_Tp, _Alloc>::_M_create_node ( _Args &&... __args ) [inline], [protected]`

## Parameters

<code>__args</code>	An instance of user data.
---------------------	---------------------------

Allocates space for a new node and constructs a copy of `__args` in it.

Definition at line 500 of file `stl_list.h`.

**4.616.3.2** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::assign ( size_type __n, const value_type & __val ) [inline]`

Assigns a given value to a list.

Parameters

<code>__n</code>	Number of elements to be assigned.
<code>__val</code>	Value to be assigned.

This function fills a list with `__n` copies of the given value. Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 702 of file `stl_list.h`.

Referenced by `std::list< __inp, __rebind_inp >::assign()`, and `std::list< __inp, __rebind_inp >::operator=()`.

**4.616.3.3** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>> void std::list<_Tp, _Alloc>::assign ( _InputIterator __first, _InputIterator __last ) [inline]`

Assigns a range to a list.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

This function fills a list with copies of the elements in the range `[__first, __last)`.

Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 721 of file `stl_list.h`.

**4.616.3.4** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::assign ( initializer_list< value_type > __l ) [inline]`

Assigns an `initializer_list` to a list.

Parameters

<code>__l</code>	An <code>initializer_list</code> of <code>value_type</code> .
------------------	---

Replace the contents of the list with copies of the elements in the `initializer_list` `__l`. This is linear in `__l.size()`.

Definition at line 743 of file `stl_list.h`.

**4.616.3.5** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::list<_Tp, _Alloc>::back ( ) [inline]`

Returns a read/write reference to the data at the last element of the list.

Definition at line 943 of file `stl_list.h`.

**4.616.3.6** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::list<_Tp, _Alloc>::back ( )  
const [inline]`

Returns a read-only (constant) reference to the data at the last element of the list.

Definition at line 955 of file `stl_list.h`.

**4.616.3.7** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::list<_Tp, _Alloc>::begin ( )  
[inline], [noexcept]`

Returns a read/write iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 758 of file `stl_list.h`.

Referenced by `std::list< __inp, __rebind_inp >::crend()`, `std::list< __inp, __rebind_inp >::front()`, `std::list< __inp, __rebind_inp >::list()`, `std::operator==( )`, `std::list< __inp, __rebind_inp >::pop_front()`, `std::list< __inp, __rebind_inp >::push_front()`, `std::list< __inp, __rebind_inp >::rend()`, `std::list< __inp, __rebind_inp >::size()`, and `std::list< __inp, __rebind_inp >::splice()`.

**4.616.3.8** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::list<_Tp, _Alloc>::begin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 767 of file `stl_list.h`.

**4.616.3.9** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::list<_Tp, _Alloc>::cbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 831 of file `stl_list.h`.

**4.616.3.10** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::list<_Tp, _Alloc>::cend ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 840 of file `stl_list.h`.

**4.616.3.11** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::clear ( ) [inline], [noexcept]`

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1228 of file `stl_list.h`.

Referenced by `std::list< __inp, __rebind_inp >::operator=( )`.

**4.616.3.12** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::list<_Tp, _Alloc>::crbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 849 of file `stl_list.h`.

**4.616.3.13** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::list<_Tp, _Alloc>::crend ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 858 of file `stl_list.h`.

**4.616.3.14** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename... _Args> iterator std::list<_Tp, _Alloc>::emplace ( iterator __position, _Args &&... __args )`

Constructs object in list before specified iterator.

#### Parameters

<code>__position</code>	A <code>const_iterator</code> into the list.
<code>__args</code>	Arguments.

#### Returns

An iterator that points to the inserted data.

This function will insert an object of type `T` constructed with `T(std::forward<Args>(args)...) before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.`

Referenced by `std::list< __inp, __rebind_inp >::insert()`.

**4.616.3.15** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> bool std::list<_Tp, _Alloc>::empty ( ) const [inline], [noexcept]`

Returns true if the list is empty. (Thus `begin()` would equal `end()`.)

Definition at line 868 of file `stl_list.h`.

Referenced by `std::list< __inp, __rebind_inp >::splice()`.

**4.616.3.16** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::list<_Tp, _Alloc>::end ( ) [inline], [noexcept]`

Returns a read/write iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 776 of file `stl_list.h`.

Referenced by `std::list< __inp, __rebind_inp >::back()`, `std::list< __inp, __rebind_inp >::cbegin()`, `std::list< __inp, __rebind_inp >::list()`, `std::operator==()`, `std::list< __inp, __rebind_inp >::push_back()`, `std::list< __inp, __rebind_inp >::rbegin()`, `std::list< __inp, __rebind_inp >::size()`, and `std::list< __inp, __rebind_inp >::splice()`.

**4.616.3.17** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::list<_Tp, _Alloc>::end ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 785 of file `stl_list.h`.

**4.616.3.18** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::list<_Tp, _Alloc>::erase ( iterator __position )`

Remove element at given position.

## Parameters

<code>__position</code>	Iterator pointing to element to be erased.
-------------------------	--

## Returns

An iterator pointing to the next element (or end()).

This function will erase the element at the given position and thus shorten the list by one.

Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Referenced by `std::list< __inp, __rebind_inp >::erase()`.

```
4.616.3.19 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::list< _Tp, _Alloc >::erase (
    iterator __first, iterator __last ) [inline]
```

Remove a range of elements.

## Parameters

<code>__first</code>	Iterator pointing to the first element to be erased.
<code>__last</code>	Iterator pointing to one past the last element to be erased.

## Returns

An iterator pointing to the element pointed to by *last* prior to erasing (or end()).

This function will erase the elements in the range [first,last) and shorten the list accordingly.

This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1193 of file `stl_list.h`.

```
4.616.3.20 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::list< _Tp, _Alloc >::front ( )
    [inline]
```

Returns a read/write reference to the data at the first element of the list.

Definition at line 927 of file `stl_list.h`.

```
4.616.3.21 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::list< _Tp, _Alloc >::front (
    ) const [inline]
```

Returns a read-only (constant) reference to the data at the first element of the list.

Definition at line 935 of file `stl_list.h`.

```
4.616.3.22 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> allocator_type std::list< _Tp, _Alloc
    >::get_allocator ( ) const [inline], [noexcept]
```

Get a copy of the memory allocation object.

Definition at line 749 of file `stl_list.h`.

Referenced by `std::list< __inp, __rebind_inp >::insert()`.

4.616.3.23 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::list<_Tp, _Alloc>::insert (`  
`iterator __position, const value_type & __x )`

Inserts given value into list before specified iterator.

## Parameters

<code>__position</code>	An iterator into the list.
<code>__x</code>	Data to be inserted.

## Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Referenced by `std::list< __inp, __rebind_inp >::insert()`.

**4.616.3.24** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::list<_Tp, _Alloc>::insert ( iterator __position, value_type && __x ) [inline]`

Inserts given rvalue into list before specified iterator.

## Parameters

<code>__position</code>	An iterator into the list.
<code>__x</code>	Data to be inserted.

## Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 1089 of file `stl_list.h`.

**4.616.3.25** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::insert ( iterator __p, initializer_list< value_type > __l ) [inline]`

Inserts the contents of an `initializer_list` into list before specified iterator.

## Parameters

<code>__p</code>	An iterator into the list.
<code>__l</code>	An <code>initializer_list</code> of <code>value_type</code> .

This function will insert copies of the data in the `initializer_list` `l` into the list before the location specified by `p`.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 1106 of file `stl_list.h`.

**4.616.3.26** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::insert ( iterator __position, size_type __n, const value_type & __x ) [inline]`

Inserts a number of copies of given data into the list.

## Parameters

---



<code>__position</code>	An iterator into the list.
<code>__n</code>	Number of elements to be inserted.
<code>__x</code>	Data to be inserted.

This function will insert a specified number of copies of the given data before the location specified by *position*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 1123 of file `stl_list.h`.

```
4.616.3.27 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator, typename =
std::_RequireInputIter<_InputIterator>> void std::list<_Tp, _Alloc>::insert ( iterator __position, _InputIterator
__first, _InputIterator __last ) [inline]
```

Inserts a range into the list.

Parameters

<code>__position</code>	An iterator into the list.
<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

This function will insert copies of the data in the range [*first,last*) into the list before the location specified by *position*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 1149 of file `stl_list.h`.

```
4.616.3.28 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::list<_Tp, _Alloc>::max_size ( )
const [inline],[noexcept]
```

Returns the `size()` of the largest possible list.

Definition at line 878 of file `stl_list.h`.

```
4.616.3.29 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::merge ( list<_Tp,
_Alloc> && __x )
```

Merge sorted lists.

Parameters

<code>__x</code>	Sorted list to merge.
------------------	-----------------------

Assumes that both `__x` and this list are sorted according to `operator<()`. Merges elements of `__x` into this list in sorted order, leaving `__x` empty when complete. Elements in this list precede elements in `__x` that are equal.

```
4.616.3.30 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _StrictWeakOrdering > void
std::list<_Tp, _Alloc>::merge ( list<_Tp, _Alloc> && __x, _StrictWeakOrdering __comp )
```

Merge sorted lists according to comparison function.

Template Parameters

<code>_StrictWeakOrdering</code>	Comparison function defining sort order.
----------------------------------	--

Parameters

<code>__x</code>	Sorted list to merge.
------------------	-----------------------

<code>__comp</code>	Comparison functor.
---------------------	---------------------

Assumes that both `__x` and this list are sorted according to `StrictWeakOrdering`. Merges elements of `__x` into this list in sorted order, leaving `__x` empty when complete. Elements in this list precede elements in `__x` that are equivalent according to `StrictWeakOrdering()`.

**4.616.3.31** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> list& std::list<_Tp, _Alloc>::operator= ( const list<_Tp, _Alloc> & __x )`

List assignment operator.

No explicit dtor needed as the `_Base` dtor takes care of things. The `_Base` dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

#### Parameters

<code>__x</code>	A list of identical element and allocator types.
------------------	--

All the elements of `__x` are copied, but unlike the copy constructor, the allocator object is not copied.

**4.616.3.32** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> list& std::list<_Tp, _Alloc>::operator= ( list<_Tp, _Alloc> && __x ) [inline]`

List move assignment operator.

#### Parameters

<code>__x</code>	A list of identical element and allocator types.
------------------	--

The contents of `__x` are moved into this list (without copying). `__x` is a valid, but unspecified list

Definition at line 667 of file `stl_list.h`.

**4.616.3.33** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> list& std::list<_Tp, _Alloc>::operator= ( initializer_list<value_type> __l ) [inline]`

List initializer list assignment operator.

#### Parameters

<code>__l</code>	An <code>initializer_list</code> of <code>value_type</code> .
------------------	---

Replace the contents of the list with copies of the elements in the `initializer_list` `__l`. This is linear in `l.size()`.

Definition at line 684 of file `stl_list.h`.

**4.616.3.34** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::pop_back ( ) [inline]`

Removes last element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before `pop_back()` is called.

Definition at line 1041 of file `stl_list.h`.

**4.616.3.35** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::pop_front ( ) [inline]`

Removes first element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in

constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop_front()` is called.

Definition at line 1001 of file `stl_list.h`.

```
4.616.3.36  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::push_back ( const
            value_type & __x )  [inline]
```

Add data to the end of the list.

Parameters

<code>__x</code>	Data to be added.
------------------	-------------------

This is a typical stack operation. The function creates an element at the end of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 1015 of file `stl_list.h`.

```
4.616.3.37  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::push_front ( const
            value_type & __x )  [inline]
```

Add data to the front of the list.

Parameters

<code>__x</code>	Data to be added.
------------------	-------------------

This is a typical stack operation. The function creates an element at the front of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 974 of file `stl_list.h`.

```
4.616.3.38  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reverse_iterator std::list<_Tp, _Alloc>
            >::rbegin ( )  [inline], [noexcept]
```

Returns a read/write reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 794 of file `stl_list.h`.

```
4.616.3.39  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::list<_Tp, _Alloc>
            >::rbegin ( ) const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 803 of file `stl_list.h`.

```
4.616.3.40  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::remove ( const _Tp
            & __value )
```

Remove all elements equal to `value`.

Parameters

<code>__value</code>	The value to remove.
----------------------	----------------------

Removes every element in the list equal to `value`. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

4.616.3.41 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _Predicate> void std::list<_Tp, _Alloc>::remove_if ( _Predicate )`

Remove all elements satisfying a predicate.

## Template Parameters

<code>__Predicate</code>	Unary predicate function or object.
--------------------------	-------------------------------------

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**4.616.3.42** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reverse_iterator std::list<_Tp, _Alloc>::rend ( ) [inline], [noexcept]`

Returns a read/write reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 812 of file `stl_list.h`.

**4.616.3.43** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::list<_Tp, _Alloc>::rend ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 821 of file `stl_list.h`.

**4.616.3.44** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::resize ( size_type __new_size )`

Resizes the list to the specified number of elements.

## Parameters

<code>__new_size</code>	Number of elements the list should contain.
-------------------------	---

This function will resize the list to the specified number of elements. If the number is smaller than the list's current size the list is truncated, otherwise default constructed elements are appended.

**4.616.3.45** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::resize ( size_type __new_size, const value_type & __x )`

Resizes the list to the specified number of elements.

## Parameters

<code>__new_size</code>	Number of elements the list should contain.
<code>__x</code>	Data with which new elements should be populated.

This function will resize the list to the specified number of elements. If the number is smaller than the list's current size the list is truncated, otherwise the list is extended and new elements are populated with given data.

**4.616.3.46** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::reverse ( ) [inline], [noexcept]`

Reverse the elements in list.

Reverse the order of elements in the list in linear time.

Definition at line 1449 of file `stl_list.h`.

**4.616.3.47** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::list<_Tp, _Alloc>::size ( ) const [inline], [noexcept]`

Returns the number of elements in the list.

Definition at line 873 of file stl\_list.h.

**4.616.3.48** template<typename \_Tp, typename \_Alloc = std::allocator<\_Tp>> void std::list<\_Tp, \_Alloc>::sort ( )

Sort the elements.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

**4.616.3.49** template<typename \_Tp, typename \_Alloc = std::allocator<\_Tp>> template<typename \_StrictWeakOrdering > void std::list<\_Tp, \_Alloc>::sort ( \_StrictWeakOrdering )

Sort the elements according to comparison function.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

**4.616.3.50** template<typename \_Tp, typename \_Alloc = std::allocator<\_Tp>> void std::list<\_Tp, \_Alloc>::splice ( iterator \_\_position, list<\_Tp, \_Alloc> && \_\_x ) [inline]

Insert contents of another list.

Parameters

<code>__position</code>	Iterator referencing the element to insert before.
<code>__x</code>	Source list.

The elements of `__x` are inserted in constant time in front of the element referenced by `__position`. `__x` becomes an empty list.

Requires this != `__x`.

Definition at line 1248 of file stl\_list.h.

Referenced by std::list< \_\_inp, \_\_rebind\_inp >::insert().

**4.616.3.51** template<typename \_Tp, typename \_Alloc = std::allocator<\_Tp>> void std::list<\_Tp, \_Alloc>::splice ( iterator \_\_position, list<\_Tp, \_Alloc> && \_\_x, iterator \_\_i ) [inline]

Insert element from another list.

Parameters

<code>__position</code>	Iterator referencing the element to insert before.
<code>__x</code>	Source list.
<code>__i</code>	Iterator referencing the element to move.

Removes the element in list `__x` referenced by `__i` and inserts it into the current list before `__position`.

Definition at line 1278 of file stl\_list.h.

**4.616.3.52** template<typename \_Tp, typename \_Alloc = std::allocator<\_Tp>> void std::list<\_Tp, \_Alloc>::splice ( iterator \_\_position, list<\_Tp, \_Alloc> && \_\_x, iterator \_\_first, iterator \_\_last ) [inline]

Insert range from another list.

Parameters

<code>__position</code>	Iterator referencing the element to insert before.
<code>__x</code>	Source list.
<code>__first</code>	Iterator referencing the start of range in x.

<code>__last</code>	Iterator referencing the end of range in <code>x</code> .
---------------------	---

Removes elements in the range `[__first,__last)` and inserts them before `__position` in constant time.

Undefined if `__position` is in `[__first,__last)`.

Definition at line 1314 of file `stl_list.h`.

**4.616.3.53** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::swap ( list<_Tp, _Alloc> & __x ) [inline]`

Swaps data with another list.

#### Parameters

<code>__x</code>	A list of the same element and allocator types.
------------------	---

This exchanges the elements between two lists in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(l1,l2)` will feed to this function.

Definition at line 1210 of file `stl_list.h`.

Referenced by `std::list< __inp, __rebind_inp >::operator=()`, and `std::swap()`.

**4.616.3.54** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::unique ( )`

Remove consecutive duplicate elements.

For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**4.616.3.55** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _BinaryPredicate > void std::list<_Tp, _Alloc>::unique ( _BinaryPredicate )`

Remove consecutive elements satisfying a predicate.

#### Template Parameters

<code>_BinaryPredicate</code>	Binary predicate function or object.
-------------------------------	--------------------------------------

For each consecutive set of elements `[first,last)` that satisfy `predicate(first,i)` where `i` is an iterator in `[first,last)`, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

The documentation for this class was generated from the following file:

- [stl\\_list.h](#)

## 4.617 std::locale Class Reference

### Classes

- class [facet](#)
- class [id](#)

### Public Types

- typedef int [category](#)

## Public Member Functions

- `locale ()` throw ()
- `locale (const locale &__other)` throw ()
- `locale (const char *__s)`
- `locale (const locale &__base, const char *__s, category __cat)`
- `locale (const locale &__base, const locale &__add, category __cat)`
- `template<typename _Facet > locale (const locale &__other, _Facet *__f)`
- `~locale ()` throw ()
- `template<typename _Facet > locale combine (const locale &__other) const`
- `string name () const`
- `bool operator!= (const locale &__other) const` throw ()
- `template<typename _Char, typename _Traits, typename _Alloc > bool operator() (const basic_string< _Char, _Traits, _Alloc > &__s1, const basic_string< _Char, _Traits, _Alloc > &__s2) const`
- `const locale & operator= (const locale &__other)` throw ()
- `bool operator== (const locale &__other) const` throw ()

## Static Public Member Functions

- `static const locale & classic ()`
- `static locale global (const locale &__loc)`

## Static Public Attributes

- `static const category none`
- `static const category ctype`
- `static const category numeric`
- `static const category collate`
- `static const category time`
- `static const category monetary`
- `static const category messages`
- `static const category all`

## Friends

- `template<typename _Cache > struct __use_cache`
- `class _Impl`
- `class facet`
- `template<typename _Facet > bool has_facet (const locale &) throw ()`
- `template<typename _Facet > const _Facet & use_facet (const locale &)`

## 4.617.1 Detailed Description

Container class for localization functionality.

The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing.

Constructing C++ locales does not change the C library locale.

This library supports efficient construction and copying of locales through a reference counting implementation of the locale class.

Definition at line 62 of file locale\_classes.h.



#### 4.617.2 Member Typedef Documentation

##### 4.617.2.1 `typedef int std::locale::category`

Definition of `locale::category`.

Definition at line 67 of file `locale_classes.h`.

#### 4.617.3 Constructor & Destructor Documentation

##### 4.617.3.1 `std::locale::locale ( ) throw`

Default constructor.

Constructs a copy of the global locale. If no locale has been explicitly set, this is the C locale.

##### 4.617.3.2 `std::locale::locale ( const locale & __other ) throw`

Copy constructor.

Constructs a copy of *other*.

###### Parameters

<code>__other</code>	The locale to copy.
----------------------	---------------------

##### 4.617.3.3 `std::locale::locale ( const char * __s ) [explicit]`

Named locale constructor.

Constructs a copy of the named C library locale.

###### Parameters

<code>__s</code>	Name of the locale to construct.
------------------	----------------------------------

###### Exceptions

<code>std::runtime_error</code>	if <code>__s</code> is null or an undefined locale.
---------------------------------	---

##### 4.617.3.4 `std::locale::locale ( const locale & __base, const char * __s, category __cat )`

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale named by *s*. If *base* is named, this locale instance will also be named.

###### Parameters

<code>__base</code>	The locale to copy.
<code>__s</code>	Name of the locale to use facets from.
<code>__cat</code>	Set of categories defining the facets to use from <code>__s</code> .

###### Exceptions

<code>std::runtime_error</code>	if <code>__s</code> is null or an undefined locale.
---------------------------------	---

#### 4.617.3.5 std::locale::locale ( const locale & \_\_base, const locale & \_\_add, category \_\_cat )

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale *add*. If *base* and *add* are named, this locale instance will also be named.

Parameters

<code>__base</code>	The locale to copy.
<code>__add</code>	The locale to use facets from.
<code>__cat</code>	Set of categories defining the facets to use from <i>add</i> .

#### 4.617.3.6 template<typename \_Facet > std::locale::locale ( const locale & \_\_other, \_Facet \* \_\_f )

Construct locale with another facet.

Constructs a copy of the locale *\_\_other*. The facet *\_\_f* is added to *\_\_other*, replacing an existing facet of type *Facet* if there is one. If *\_\_f* is null, this locale is a copy of *\_\_other*.

Parameters

<code>__other</code>	The locale to copy.
<code>__f</code>	The facet to add in.

#### 4.617.3.7 std::locale::~~locale ( ) throw

Locale destructor.

### 4.617.4 Member Function Documentation

#### 4.617.4.1 static const locale& std::locale::classic ( ) [static]

Return reference to the C locale.

#### 4.617.4.2 template<typename \_Facet > locale std::locale::combine ( const locale & \_\_other ) const

Construct locale with another facet.

Constructs and returns a new copy of this locale. Adds or replaces an existing facet of type *Facet* from the locale *other* into the new locale.

Template Parameters

<code>_Facet</code>	The facet type to copy from <i>other</i>
---------------------	--

Parameters

<code>__other</code>	The locale to copy from.
----------------------	--------------------------

Returns

Newly constructed locale.

## Exceptions

<code>std::runtime_error</code>	if <code>__other</code> has no facet of type <code>_Facet</code> .
---------------------------------	--

4.617.4.3 `static locale std::locale::global ( const locale & __loc ) [static]`

Set global locale.

This function sets the global locale to the argument and returns a copy of the previous global locale. If the argument has a name, it will also call `std::setlocale(LC_ALL, loc.name())`.

## Parameters

<code>__loc</code>	The new locale to make global.
--------------------	--------------------------------

## Returns

Copy of the old global locale.

4.617.4.4 `string std::locale::name ( ) const`

Return locale name.

## Returns

Locale name or "\*" if unnamed.

4.617.4.5 `bool std::locale::operator!=( const locale & __other ) const throw () [inline]`

Locale inequality.

## Parameters

<code>__other</code>	The locale to compare against.
----------------------	--------------------------------

## Returns

`! (*this == __other)`

Definition at line 235 of file `locale_classes.h`.

References `operator==( )`.

4.617.4.6 `template<typename _Char, typename _Traits, typename _Alloc> bool std::locale::operator() ( const basic_string<_Char, _Traits, _Alloc> & __s1, const basic_string<_Char, _Traits, _Alloc> & __s2 ) const`

Compare two strings according to collate.

Template operator to compare two strings using the compare function of the collate facet in this locale. One use is to provide the locale to the sort function. For example, a vector `v` of strings could be sorted according to locale `loc` by doing:

```
std::sort(v.begin(), v.end(), loc);
```

## Parameters

<code>__s1</code>	First string to compare.
<code>__s2</code>	Second string to compare.

## Returns

True if `collate<_Char> facet` compares `__s1 < __s2`, else false.

**4.617.4.7 const locale& std::locale::operator= ( const locale & \_\_other ) throw )**

Assignment operator.

Set this locale to be a copy of *other*.

## Parameters

<code>__other</code>	The locale to copy.
----------------------	---------------------

## Returns

A reference to this locale.

**4.617.4.8 bool std::locale::operator== ( const locale & \_\_other ) const throw )**

Locale equality.

## Parameters

<code>__other</code>	The locale to compare against.
----------------------	--------------------------------

## Returns

True if *other* and this refer to the same locale instance, are copies, or have the same name. False otherwise.

Referenced by `operator!=()`.

**4.617.5 Member Data Documentation****4.617.5.1 const category std::locale::all [static]**

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 105 of file `locale_classes.h`.

**4.617.5.2 const category std::locale::collate [static]**

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 101 of file `locale_classes.h`.

#### 4.617.5.3 `const category std::locale::ctype` `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 99 of file `locale_classes.h`.

#### 4.617.5.4 `const category std::locale::messages` `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 104 of file `locale_classes.h`.

#### 4.617.5.5 `const category std::locale::monetary` `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 103 of file `locale_classes.h`.

#### 4.617.5.6 `const category std::locale::none` `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 98 of file `locale_classes.h`.

#### 4.617.5.7 `const category std::locale::numeric` `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 100 of file `locale_classes.h`.

#### 4.617.5.8 `const category std::locale::time` `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

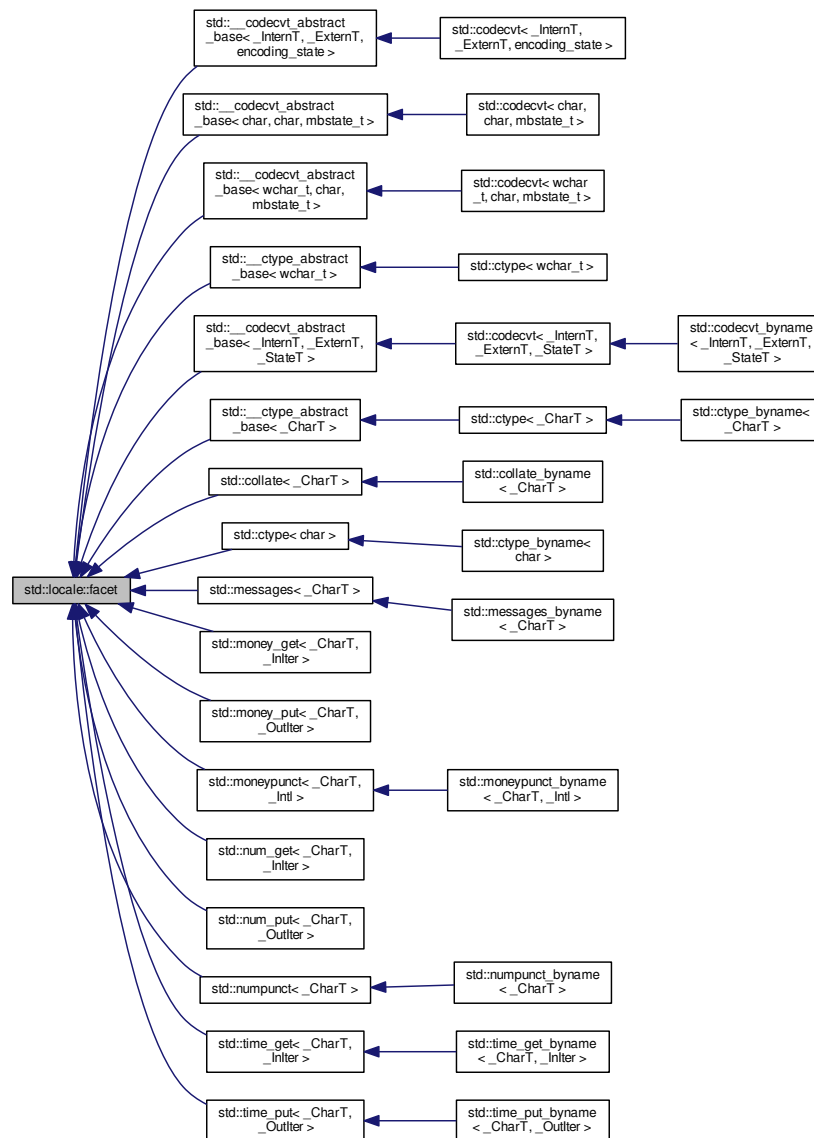
Definition at line 102 of file `locale_classes.h`.

The documentation for this class was generated from the following file:

- [locale\\_classes.h](#)

## 4.618 std::locale::facet Class Reference

Inheritance diagram for std::locale::facet:



### Protected Member Functions

- [facet](#) (size\_t \_\_refs=0) throw ()
- virtual [~facet](#) ()

### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

### Friends

- class `locale`
- class `locale::_Impl`

#### 4.618.1 Detailed Description

Localization functionality base class.

The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.

Facets may not be copied or assigned.

Definition at line 338 of file `locale_classes.h`.

#### 4.618.2 Constructor & Destructor Documentation

4.618.2.1 `std::locale::facet::facet ( size_t __refs = 0 ) throw ()` `[inline]`, `[explicit]`, `[protected]`

Facet constructor.

This is the constructor provided by the standard. If `refs` is 0, the facet is destroyed when the last referencing locale is destroyed. Otherwise the facet will never be destroyed.

##### Parameters

<code>__refs</code>	The initial value for reference count.
---------------------	--

Definition at line 370 of file `locale_classes.h`.

4.618.2.2 `virtual std::locale::facet::~facet ( )` `[protected]`, `[virtual]`

Facet destructor.

The documentation for this class was generated from the following file:

- [locale\\_classes.h](#)

#### 4.619 std::locale::id Class Reference

##### Public Member Functions

- `id ()`
- `size_t _M_id () const throw ()`

## Friends

- template<typename \_Facet > bool **has\_facet** (const [locale](#) &) throw ()
- class **locale**
- class **locale::Impl**
- template<typename \_Facet > const \_Facet & **use\_facet** (const [locale](#) &)

## 4.619.1 Detailed Description

Facet ID class.

The ID class provides facets with an index used to identify them. Every facet class must define a public static member locale::id, or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The locale::id ensures that each class type gets a unique identifier.

Definition at line 436 of file locale\_classes.h.

## 4.619.2 Constructor &amp; Destructor Documentation

## 4.619.2.1 std::locale::id::id( ) [inline]

Constructor.

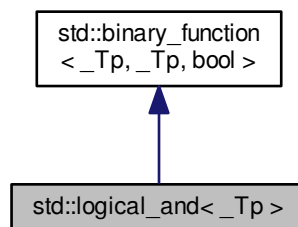
Definition at line 467 of file locale\_classes.h.

The documentation for this class was generated from the following file:

- [locale\\_classes.h](#)

## 4.620 std::logical\_and&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::logical\_and< \_Tp >:



## Public Types

- typedef \_Tp [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef \_Tp [second\\_argument\\_type](#)



## Public Member Functions

- **bool operator()** (const \_Tp &\_\_x, const \_Tp &\_\_y) const

### 4.620.1 Detailed Description

template<typename \_Tp>struct std::logical\_and< \_Tp >

One of the [Boolean operations functors](#).

Definition at line 268 of file stl\_function.h.

### 4.620.2 Member Typedef Documentation

4.620.2.1 typedef \_Tp std::binary\_function< \_Tp, \_Tp, bool >::first\_argument\_type [inherited]

first\_argument\_type is the type of the first argument

Definition at line 117 of file stl\_function.h.

4.620.2.2 typedef bool std::binary\_function< \_Tp, \_Tp, bool >::result\_type [inherited]

result\_type is the return type

Definition at line 123 of file stl\_function.h.

4.620.2.3 typedef \_Tp std::binary\_function< \_Tp, \_Tp, bool >::second\_argument\_type [inherited]

second\_argument\_type is the type of the second argument

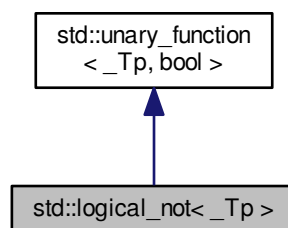
Definition at line 120 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.621 std::logical\_not< \_Tp > Struct Template Reference

Inheritance diagram for std::logical\_not< \_Tp >:



## Public Types

- typedef `_Tp` [argument\\_type](#)
- typedef `bool` [result\\_type](#)

## Public Member Functions

- `bool` **operator()** (`const _Tp &__x`) `const`

## 4.621.1 Detailed Description

`template<typename _Tp>struct std::logical_not< _Tp >`

One of the [Boolean operations functors](#).

Definition at line 286 of file `stl_function.h`.

## 4.621.2 Member Typedef Documentation

4.621.2.1 `typedef _Tp std::unary_function< _Tp, bool >::argument_type` [\[inherited\]](#)

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

4.621.2.2 `typedef bool std::unary_function< _Tp, bool >::result_type` [\[inherited\]](#)

`result_type` is the return type

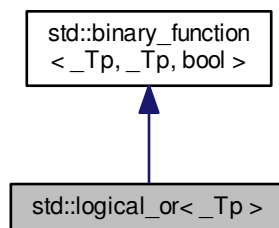
Definition at line 107 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.622 std::logical\_or&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for `std::logical_or< _Tp >`:



**Public Types**

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

**Public Member Functions**

- `bool` **operator()** (`const _Tp &__x`, `const _Tp &__y`) `const`

**4.622.1 Detailed Description**

`template<typename _Tp>struct std::logical_or<_Tp>`

One of the [Boolean operations functors](#).

Definition at line 277 of file `stl_function.h`.

**4.622.2 Member Typedef Documentation**

**4.622.2.1** `typedef _Tp std::binary_function<_Tp, _Tp, bool>::first_argument_type` `[inherited]`

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

**4.622.2.2** `typedef bool std::binary_function<_Tp, _Tp, bool>::result_type` `[inherited]`

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

**4.622.2.3** `typedef _Tp std::binary_function<_Tp, _Tp, bool>::second_argument_type` `[inherited]`

`second_argument_type` is the type of the second argument

Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

**4.623 std::lognormal\_distribution<\_RealType> Class Template Reference****Classes**

- struct [param\\_type](#)

**Public Types**

- typedef `_RealType` [result\\_type](#)

## Public Member Functions

- **lognormal\_distribution** (\_RealType \_\_m=\_RealType(0), \_RealType \_\_s=\_RealType(1))
- **lognormal\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator> void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator> void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator> void **generate** ([result\\_type](#) \* \_\_f, [result\\_type](#) \* \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- \_RealType **m** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator> [result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator> [result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()
- \_RealType **s** () const

## Friends

- template<typename \_RealType1, typename \_CharT, typename \_Traits> std::basic\_ostream<\_CharT, \_Traits> & **operator<<** (std::basic\_ostream<\_CharT, \_Traits> &\_\_os, const [std::lognormal\\_distribution](#)<\_RealType1> &\_\_x)
- bool **operator==** (const [lognormal\\_distribution](#) &\_\_d1, const [lognormal\\_distribution](#) &\_\_d2)
- template<typename \_RealType1, typename \_CharT, typename \_Traits> std::basic\_istream<\_CharT, \_Traits> & **operator>>** (std::basic\_istream<\_CharT, \_Traits> &\_\_is, [std::lognormal\\_distribution](#)<\_RealType1> &\_\_x)

## 4.623.1 Detailed Description

template<typename \_RealType = double>class std::lognormal\_distribution<\_RealType>

A lognormal\_distribution random number distribution.

The formula for the normal probability mass function is

$$p(x|m, s) = \frac{1}{sx\sqrt{2\pi}} \exp - \frac{(\ln x - m)^2}{2s^2}$$

Definition at line 2298 of file random.h.

## 4.623.2 Member Typedef Documentation

4.623.2.1 template<typename \_RealType = double> typedef \_RealType std::lognormal\_distribution<\_RealType>::[result\\_type](#)

The type of the range of the distribution.

Definition at line 2301 of file random.h.

#### 4.623.3 Member Function Documentation

**4.623.3.1** `template<typename _RealType = double> result_type std::lognormal_distribution< _RealType >::max ( )  
const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 2389 of file random.h.

References std::max().

**4.623.3.2** `template<typename _RealType = double> result_type std::lognormal_distribution< _RealType >::min ( ) const  
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2382 of file random.h.

**4.623.3.3** `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type  
std::lognormal_distribution< _RealType >::operator() ( _UniformRandomNumberGenerator & __urng )  
[inline]`

Generating functions.

Definition at line 2397 of file random.h.

**4.623.3.4** `template<typename _RealType = double> param_type std::lognormal_distribution< _RealType >::param ( )  
const [inline]`

Returns the parameter set of the distribution.

Definition at line 2367 of file random.h.

**4.623.3.5** `template<typename _RealType = double> void std::lognormal_distribution< _RealType >::param ( const  
param_type & __param ) [inline]`

Sets the parameter set of the distribution.

##### Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 2375 of file random.h.

**4.623.3.6** `template<typename _RealType = double> void std::lognormal_distribution< _RealType >::reset ( )  
[inline]`

Resets the distribution state.

Definition at line 2349 of file random.h.

References std::normal\_distribution< \_RealType >::reset().

#### 4.623.4 Friends And Related Function Documentation

**4.623.4.1** `template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits  
> std::basic_ostream< _CharT , _Traits > & operator<< ( std::basic_ostream< _CharT , _Traits > & __os, const  
std::lognormal_distribution< _RealType1 > & __x ) [friend]`

Inserts a lognormal\_distribution random number distribution `__x` into the output stream `__os`.

## Parameters

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

## Returns

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

4.623.4.2 `template<typename _RealType = double> bool operator==( const lognormal_distribution<_RealType> &__d1, const lognormal_distribution<_RealType> &__d2 ) [friend]`

Return true if two lognormal distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2434 of file random.h.

4.623.4.3 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<_CharT, _Traits> &__is, std::lognormal_distribution<_RealType1> &__x ) [friend]`

Extracts a lognormal\_distribution random number distribution `__x` from the input stream `__is`.

## Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A lognormal_distribution random number generator engine.

## Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

## 4.624 std::lognormal\_distribution&lt;\_RealType&gt;::param\_type Struct Reference

## Public Types

- typedef [lognormal\\_distribution](#)<\_RealType> **distribution\_type**

## Public Member Functions

- **param\_type** (`_RealType __m=_RealType(0), _RealType __s=_RealType(1)`)
- `_RealType m` () const
- `_RealType s` () const

## Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

## 4.624.1 Detailed Description

```
template<typename _RealType = double>struct std::lognormal_distribution< _RealType >::param_type
```

Parameter type.

Definition at line 2307 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 4.625 std::map&lt; \_Key, \_Tp, \_Compare, \_Alloc &gt; Class Template Reference

## Public Types

- typedef \_Alloc **allocator\_type**
- typedef \_Rep\_type::const\_iterator **const\_iterator**
- typedef \_Pair\_alloc\_type::const\_pointer **const\_pointer**
- typedef \_Pair\_alloc\_type::const\_reference **const\_reference**
- typedef [\\_Rep\\_type::const\\_reverse\\_iterator](#) **const\_reverse\_iterator**
- typedef \_Rep\_type::difference\_type **difference\_type**
- typedef \_Rep\_type::iterator **iterator**
- typedef \_Compare **key\_compare**
- typedef \_Key **key\_type**
- typedef \_Tp **mapped\_type**
- typedef \_Pair\_alloc\_type::pointer **pointer**
- typedef \_Pair\_alloc\_type::reference **reference**
- typedef [\\_Rep\\_type::reverse\\_iterator](#) **reverse\_iterator**
- typedef \_Rep\_type::size\_type **size\_type**
- typedef [std::pair](#)< const \_Key, \_Tp > **value\_type**

## Public Member Functions

- [map](#) ()
- [map](#) (const \_Compare &\_\_comp, const allocator\_type &\_\_a=allocator\_type())
- [map](#) (const [map](#) &\_\_x)
- [map](#) ([map](#) &&\_\_x) noexcept(is\_nothrow\_copy\_constructible< \_Compare >::value)
- [map](#) (initializer\_list< [value\\_type](#) > \_\_l, const \_Compare &\_\_comp=\_Compare(), const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator > [map](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_InputIterator > [map](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_comp, const allocator\_type &\_\_a=allocator\_type())
- mapped\_type & [at](#) (const key\_type &\_\_k)
- const mapped\_type & [at](#) (const key\_type &\_\_k) const
- iterator [begin](#) () noexcept
- const\_iterator [begin](#) () const noexcept
- const\_iterator [cbegin](#) () const noexcept
- const\_iterator [cend](#) () const noexcept
- void [clear](#) () noexcept
- size\_type [count](#) (const key\_type &\_\_x) const

- [const\\_reverse\\_iterator crbegin](#) () const noexcept
- [const\\_reverse\\_iterator crend](#) () const noexcept
- [template<typename... \\_Args> std::pair< iterator, bool > \*\*emplace\*\*](#) ( \_Args &&... \_\_args)
- [template<typename... \\_Args> iterator \*\*emplace\\_hint\*\*](#) (const\_iterator \_\_pos, \_Args &&... \_\_args)
- [bool \*\*empty\*\*](#) () const noexcept
- [iterator \*\*end\*\*](#) () noexcept
- [const\\_iterator \*\*end\*\*](#) () const noexcept
- [std::pair< iterator, iterator > \*\*equal\\_range\*\*](#) (const key\_type &\_\_x)
- [std::pair< const\\_iterator, const\\_iterator > \*\*equal\\_range\*\*](#) (const key\_type &\_\_x) const
- [iterator \*\*erase\*\*](#) (const\_iterator \_\_position)
- [\\_GLIBCXX\\_ABI\\_TAG\\_CXX11 iterator \*\*erase\*\*](#) (iterator \_\_position)
- [size\\_type \*\*erase\*\*](#) (const key\_type &\_\_x)
- [iterator \*\*erase\*\*](#) (const\_iterator \_\_first, const\_iterator \_\_last)
- [iterator \*\*find\*\*](#) (const key\_type &\_\_x)
- [const\\_iterator \*\*find\*\*](#) (const key\_type &\_\_x) const
- [allocator\\_type \*\*get\\_allocator\*\*](#) () const noexcept
- [std::pair< iterator, bool > \*\*insert\*\*](#) (const [value\\_type](#) &\_\_x)
- [template<typename \\_Pair, typename = typename std::enable\\_if<std::is\\_constructible<value\\_type, \\_Pair&&>::value>::type> std::pair< iterator, bool > \*\*insert\*\*](#) ( \_Pair &&\_\_x)
- [void \*\*insert\*\*](#) (std::initializer\_list< [value\\_type](#) > \_\_list)
- [iterator \*\*insert\*\*](#) (const\_iterator \_\_position, const [value\\_type](#) &\_\_x)
- [template<typename \\_Pair, typename = typename std::enable\\_if<std::is\\_constructible<value\\_type, \\_Pair&&>::value>::type> iterator \*\*insert\*\*](#) (const\_iterator \_\_position, \_Pair &&\_\_x)
- [template<typename \\_InputIterator > void \*\*insert\*\*](#) ( \_InputIterator \_\_first, \_InputIterator \_\_last)
- [key\\_compare \*\*key\\_comp\*\*](#) () const
- [iterator \*\*lower\\_bound\*\*](#) (const key\_type &\_\_x)
- [const\\_iterator \*\*lower\\_bound\*\*](#) (const key\_type &\_\_x) const
- [size\\_type \*\*max\\_size\*\*](#) () const noexcept
- [map & \*\*operator=\*\*](#) (const [map](#) &\_\_x)
- [map & \*\*operator=\*\*](#) ( [map](#) &&\_\_x)
- [map & \*\*operator=\*\*](#) (initializer\_list< [value\\_type](#) > \_\_l)
- [mapped\\_type & \*\*operator\[\]\*\*](#) (const key\_type &\_\_k)
- [mapped\\_type & \*\*operator\[\]\*\*](#) (key\_type &&\_\_k)
- [reverse\\_iterator \*\*rbegin\*\*](#) () noexcept
- [const\\_reverse\\_iterator \*\*rbegin\*\*](#) () const noexcept
- [reverse\\_iterator \*\*rend\*\*](#) () noexcept
- [const\\_reverse\\_iterator \*\*rend\*\*](#) () const noexcept
- [size\\_type \*\*size\*\*](#) () const noexcept
- [void \*\*swap\*\*](#) ( [map](#) &\_\_x)
- [iterator \*\*upper\\_bound\*\*](#) (const key\_type &\_\_x)
- [const\\_iterator \*\*upper\\_bound\*\*](#) (const key\_type &\_\_x) const
- [value\\_compare \*\*value\\_comp\*\*](#) () const

## Friends

- [template<typename \\_K1, typename \\_T1, typename \\_C1, typename \\_A1 > bool \*\*operator<\*\*](#) (const [map](#)< \_K1, \_T1, \_C1, \_A1 > &, const [map](#)< \_K1, \_T1, \_C1, \_A1 > &)
- [template<typename \\_K1, typename \\_T1, typename \\_C1, typename \\_A1 > bool \*\*operator==\*\*](#) (const [map](#)< \_K1, \_T1, \_C1, \_A1 > &, const [map](#)< \_K1, \_T1, \_C1, \_A1 > &)



#### 4.625.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> class std::map< _Key, _Tp, _Compare, _Alloc >
```

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

## Template Parameters

<code>_Key</code>	Type of key objects.
<code>_Tp</code>	Type of mapped objects.
<code>_Compare</code>	Comparison function object type, defaults to <code>less&lt;_Key&gt;</code> .
<code>_Alloc</code>	Allocator type, defaults to <code>allocator&lt;pair&lt;const _Key, _Tp&gt;&gt;</code> .

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using unique keys). For a `map<Key, T>` the `key_type` is `Key`, the `mapped_type` is `T`, and the `value_type` is `std::pair<const Key, T>`.

Maps support bidirectional iterators.

The private tree data is declared exactly the same way for `map` and `multimap`; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 96 of file `stl_map.h`.

## 4.625.2 Constructor &amp; Destructor Documentation

4.625.2.1 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::map<_Key, _Tp, _Compare, _Alloc>::map ( ) [inline]`

Default constructor creates no elements.

Definition at line 160 of file `stl_map.h`.

4.625.2.2 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::map<_Key, _Tp, _Compare, _Alloc>::map ( const _Compare & __comp, const allocator_type & __a = allocator_type() ) [inline], [explicit]`

Creates a map with no elements.

## Parameters

<code>__comp</code>	A comparison object.
<code>__a</code>	An allocator object.

Definition at line 169 of file `stl_map.h`.

4.625.2.3 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::map<_Key, _Tp, _Compare, _Alloc>::map ( const map<_Key, _Tp, _Compare, _Alloc> & __x ) [inline]`

Map copy constructor.

## Parameters

<code>__x</code>	A map of identical element and allocator types.
------------------	---

The newly-created map uses a copy of the allocation object used by `__x`.

Definition at line 180 of file `stl_map.h`.

4.625.2.4 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::map<_Key, _Tp, _Compare, _Alloc>::map ( map<_Key, _Tp, _Compare, _Alloc> && __x ) [inline], [noexcept]`

Map move constructor.

## Parameters

<code>__x</code>	A map of identical element and allocator types.
------------------	---

The newly-created map contains the exact contents of `__x`. The contents of `__x` are a valid, but unspecified map.

Definition at line 191 of file `stl_map.h`.

```
4.625.2.5  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp> >> std::map<_Key, _Tp, _Compare, _Alloc >::map (
            initializer_list< value_type > __l, const _Compare & __comp = _Compare(), const allocator_type & __a =
            allocator_type() ) [inline]
```

Builds a map from an `initializer_list`.

## Parameters

<code>__l</code>	An <code>initializer_list</code> .
<code>__comp</code>	A comparison object.
<code>__a</code>	An allocator object.

Create a map consisting of copies of the elements in the `initializer_list` `__l`. This is linear in N if the range is already sorted, and NlogN otherwise (where N is `__l.size()`).

Definition at line 206 of file `stl_map.h`.

```
4.625.2.6  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp> >> template<typename _InputIterator > std::map<_Key, _Tp, _Compare,
            _Alloc >::map ( _InputIterator __first, _InputIterator __last ) [inline]
```

Builds a map from a range.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

Create a map consisting of copies of the elements from `[__first,__last)`. This is linear in N if the range is already sorted, and NlogN otherwise (where N is `distance(__first,__last)`).

Definition at line 224 of file `stl_map.h`.

```
4.625.2.7  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp> >> template<typename _InputIterator > std::map<_Key, _Tp, _Compare,
            _Alloc >::map ( _InputIterator __first, _InputIterator __last, const _Compare & __comp, const allocator_type & __a =
            allocator_type() ) [inline]
```

Builds a map from a range.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__comp</code>	A comparison functor.
<code>__a</code>	An allocator object.

Create a map consisting of copies of the elements from `[__first,__last)`. This is linear in N if the range is already sorted, and NlogN otherwise (where N is `distance(__first,__last)`).

Definition at line 241 of file `stl_map.h`.

## 4.625.3 Member Function Documentation

```
4.625.3.1 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =  
std::allocator<std::pair<const _Key, _Tp> >> mapped_type& std::map<_Key, _Tp, _Compare, _Alloc >::at ( const  
key_type & __k ) [inline]
```

Access to map data.

## Parameters

<code>__k</code>	The key for which data should be retrieved.
------------------	---

## Returns

A reference to the data whose key is equivalent to `__k`, if such a data is present in the map.

## Exceptions

<code>std::out_of_range</code>	If no such data is present.
--------------------------------	-----------------------------

Definition at line 501 of file `stl_map.h`.

References `std::map<_Key, _Tp, _Compare, _Alloc>::end()`, `std::map<_Key, _Tp, _Compare, _Alloc>::key_comp()`, and `std::map<_Key, _Tp, _Compare, _Alloc>::lower_bound()`.

**4.625.3.2** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::map<_Key, _Tp, _Compare, _Alloc>::begin ( )`  
`[inline], [noexcept]`

Returns a read/write iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 320 of file `stl_map.h`.

**4.625.3.3** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::map<_Key, _Tp, _Compare, _Alloc>::begin ( )`  
`const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 329 of file `stl_map.h`.

**4.625.3.4** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::map<_Key, _Tp, _Compare, _Alloc>::cbegin ( )`  
`const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 393 of file `stl_map.h`.

**4.625.3.5** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::map<_Key, _Tp, _Compare, _Alloc>::cend ( )`  
`const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 402 of file `stl_map.h`.

**4.625.3.6** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> void std::map<_Key, _Tp, _Compare, _Alloc>::clear ( )`  
`[inline], [noexcept]`

Erases all elements in a map. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 788 of file stl\_map.h.

Referenced by std::map< \_Key, \_Tp, \_Compare, \_Alloc >::operator=().

**4.625.3.7** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =  
std::allocator<std::pair<const _Key, _Tp>>> size_type std::map<_Key, _Tp, _Compare, _Alloc>::count ( const  
key_type & __x ) const [inline]`

Finds the number of elements with given key.

Parameters

<code>__x</code>	Key of (key, value) pairs to be located.
------------------	--

Returns

Number of elements with specified key.

This function only makes sense for multimaps; for map the result will either be 0 (not present) or 1 (present).

Definition at line 848 of file stl\_map.h.

**4.625.3.8** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =  
std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::map<_Key, _Tp, _Compare, _Alloc  
>::crbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 411 of file stl\_map.h.

**4.625.3.9** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =  
std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::map<_Key, _Tp, _Compare, _Alloc  
>::crend ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 420 of file stl\_map.h.

**4.625.3.10** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =  
std::allocator<std::pair<const _Key, _Tp>>> template<typename... _Args> std::pair<iterator, bool>  
std::map<_Key, _Tp, _Compare, _Alloc>::emplace ( _Args &&... __args ) [inline]`

Attempts to build and insert a std::pair into the map.

Parameters

<code>__args</code>	Arguments used to generate a new pair instance (see std::piecewise_construct for passing arguments to each part of the pair constructor).
---------------------	---

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to build and insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.

Insertion requires logarithmic time.

Definition at line 540 of file `stl_map.h`.

```
4.625.3.11 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> template<typename... _Args> iterator std::map<_Key, _Tp,
_Compare, _Alloc >::emplace_hint ( const_iterator __pos, _Args &&... __args ) [inline]
```

Attempts to build and insert a `std::pair` into the map.

#### Parameters

<code>__pos</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__args</code>	Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor).

#### Returns

An iterator that points to the element with key of the `std::pair` built from `__args` (may or may not be that `std::pair`).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `emplace()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 570 of file `stl_map.h`.

```
4.625.3.12 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> bool std::map<_Key, _Tp, _Compare, _Alloc >::empty ( ) const
[inline], [noexcept]
```

Returns true if the map is empty. (Thus `begin()` would equal `end()`.)

Definition at line 429 of file `stl_map.h`.

```
4.625.3.13 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::map<_Key, _Tp, _Compare, _Alloc >::end ( )
[inline], [noexcept]
```

Returns a read/write iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 338 of file `stl_map.h`.

Referenced by `std::map<_Key, _Tp, _Compare, _Alloc >::at()`, and `std::map<_Key, _Tp, _Compare, _Alloc >::operator[]()`.

```
4.625.3.14 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::map<_Key, _Tp, _Compare, _Alloc >::end ( )
const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 347 of file `stl_map.h`.

```
4.625.3.15 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =  
std::allocator<std::pair<const _Key, _Tp> >> std::pair<iterator, iterator> std::map<_Key, _Tp, _Compare,  
_Alloc >::equal_range ( const key_type & __x ) [inline]
```

Finds a subsequence matching given key.



**Parameters**

<code>__x</code>	Key of (key, value) pairs to be located.
------------------	--

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

Definition at line 917 of file stl\_map.h.

```
4.625.3.16 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> std::pair<const_iterator, const_iterator> std::map<_Key, _Tp,
_Compare, _Alloc >::equal_range ( const key_type & __x ) const    [inline]
```

Finds a subsequence matching given key.

**Parameters**

<code>__x</code>	Key of (key, value) pairs to be located.
------------------	--

**Returns**

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

Definition at line 936 of file stl\_map.h.

```
4.625.3.17 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> iterator std::map<_Key, _Tp, _Compare, _Alloc >::erase (
const_iterator __position )    [inline]
```

Erases an element from a map.

**Parameters**

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

**Returns**

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 690 of file `stl_map.h`.

```
4.625.3.18 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> size_type std::map< _Key, _Tp, _Compare, _Alloc >::erase ( const
key_type & __x ) [inline]
```

Erases elements according to the provided key.

#### Parameters

<code>__x</code>	Key of element to be erased.
------------------	------------------------------

#### Returns

The number of elements erased.

This function erases all the elements located by the given key from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 726 of file `stl_map.h`.

```
4.625.3.19 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::map< _Key, _Tp, _Compare, _Alloc >::erase (
const_iterator __first, const_iterator __last ) [inline]
```

Erases a `[first,last)` range of elements from a map.

#### Parameters

<code>__first</code>	Iterator pointing to the start of the range to be erased.
<code>__last</code>	Iterator pointing to the end of the range to be erased.

#### Returns

The iterator `__last`.

This function erases a sequence of elements from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 746 of file `stl_map.h`.

```
4.625.3.20 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::map< _Key, _Tp, _Compare, _Alloc >::find ( const
key_type & __x ) [inline]
```

Tries to locate an element in a map.

#### Parameters

<code>__x</code>	Key of (key, value) pair to be located.
------------------	---

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 821 of file `stl_map.h`.

```
4.625.3.21  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::map<_Key, _Tp, _Compare, _Alloc>::find (
            const key_type & __x ) const    [inline]
```

Tries to locate an element in a map.

**Parameters**

<code>__x</code>	Key of (key, value) pair to be located.
------------------	---

**Returns**

Read-only (constant) iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 836 of file `stl_map.h`.

```
4.625.3.22  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> allocator_type std::map<_Key, _Tp, _Compare, _Alloc>
            >::get_allocator ( ) const    [inline], [noexcept]
```

Get a copy of the memory allocation object.

Definition at line 310 of file `stl_map.h`.

```
4.625.3.23  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> std::pair<iterator, bool> std::map<_Key, _Tp, _Compare, _Alloc>
            >::insert ( const value_type & __x )    [inline]
```

Attempts to insert a `std::pair` into the map.

**Parameters**

<code>__x</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).
------------------	---

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.

Insertion requires logarithmic time.

Definition at line 594 of file `stl_map.h`.

Referenced by std::map< \_Key, \_Tp, \_Compare, \_Alloc >::insert(), std::map< \_Key, \_Tp, \_Compare, \_Alloc >::operator=(), and std::map< \_Key, \_Tp, \_Compare, \_Alloc >::operator[]().

4.625.3.24 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> void std::map< _Key, _Tp, _Compare, _Alloc >::insert ( std::initializer_list< value_type > __list ) [inline]`

Attempts to insert a list of std::pairs into the map.

#### Parameters

<code>__list</code>	A std::initializer_list<value_type> of pairs to be inserted.
---------------------	--

Complexity similar to that of the range constructor.

Definition at line 615 of file stl\_map.h.

References std::map< \_Key, \_Tp, \_Compare, \_Alloc >::insert().

4.625.3.25 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::map< _Key, _Tp, _Compare, _Alloc >::insert ( const_iterator __position, const value_type & __x ) [inline]`

Attempts to insert a std::pair into the map.

#### Parameters

<code>__position</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__x</code>	Pair to be inserted (see std::make_pair for easy creation of pairs).

#### Returns

An iterator that points to the element with key of \_\_x (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 644 of file stl\_map.h.

4.625.3.26 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> template<typename _InputIterator > void std::map< _Key, _Tp, _Compare, _Alloc >::insert ( _InputIterator __first, _InputIterator __last ) [inline]`

Template function that attempts to insert a range of elements.

#### Parameters

<code>__first</code>	Iterator pointing to the start of the range to be inserted.
<code>__last</code>	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 670 of file stl\_map.h.

```
4.625.3.27 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> key_compare std::map< _Key, _Tp, _Compare, _Alloc >::key_comp (
) const [inline]
```

Returns the key comparison object out of which the map was constructed.

Definition at line 797 of file stl\_map.h.

Referenced by std::map< \_Key, \_Tp, \_Compare, \_Alloc >::at(), and std::map< \_Key, \_Tp, \_Compare, \_Alloc >::operator[]().

```
4.625.3.28 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::map< _Key, _Tp, _Compare, _Alloc >::lower_bound (
const key_type & __x ) [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

__x	Key of (key, value) pair to be located.
-----	---

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 863 of file stl\_map.h.

Referenced by std::map< \_Key, \_Tp, \_Compare, \_Alloc >::at(), and std::map< \_Key, \_Tp, \_Compare, \_Alloc >::operator[]().

```
4.625.3.29 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::map< _Key, _Tp, _Compare, _Alloc
>::lower_bound ( const key_type & __x ) const [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

__x	Key of (key, value) pair to be located.
-----	---

Returns

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 878 of file stl\_map.h.

```
4.625.3.30 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> size_type std::map< _Key, _Tp, _Compare, _Alloc >::max_size ( )
const [inline], [noexcept]
```

Returns the maximum size of the map.

Definition at line 439 of file stl\_map.h.

```
4.625.3.31 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> map& std::map< _Key, _Tp, _Compare, _Alloc >::operator= ( const
map< _Key, _Tp, _Compare, _Alloc > &__x ) [inline]
```

Map assignment operator.

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

#### Parameters

<code>__x</code>	A map of identical element and allocator types.
------------------	---

All the elements of `__x` are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 264 of file `stl_map.h`.

```
4.625.3.32 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> map& std::map< _Key, _Tp, _Compare, _Alloc >::operator= (
map< _Key, _Tp, _Compare, _Alloc > &&__x ) [inline]
```

Map move assignment operator.

#### Parameters

<code>__x</code>	A map of identical element and allocator types.
------------------	---

The contents of `__x` are moved into this map (without copying). `__x` is a valid, but unspecified map.

Definition at line 279 of file `stl_map.h`.

References `std::map< _Key, _Tp, _Compare, _Alloc >::clear()`, and `std::map< _Key, _Tp, _Compare, _Alloc >::swap()`.

```
4.625.3.33 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> map& std::map< _Key, _Tp, _Compare, _Alloc >::operator= (
initializer_list< value_type > __l ) [inline]
```

Map list assignment operator.

#### Parameters

<code>__l</code>	An <code>initializer_list</code> .
------------------	------------------------------------

This function fills a map with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the map and that the resulting map's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 300 of file `stl_map.h`.

References `std::map< _Key, _Tp, _Compare, _Alloc >::clear()`, and `std::map< _Key, _Tp, _Compare, _Alloc >::insert()`.

```
4.625.3.34 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> mapped_type& std::map< _Key, _Tp, _Compare, _Alloc
>::operator[] ( const key_type &__k ) [inline]
```

Subscript ( `[]` ) access to map data.

#### Parameters

<code>__k</code>	The key for which data should be retrieved.
------------------	---

**Returns**

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ( [] ) operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.

Lookup requires logarithmic time.

Definition at line 456 of file stl\_map.h.

References std::map< \_Key, \_Tp, \_Compare, \_Alloc >::end(), std::map< \_Key, \_Tp, \_Compare, \_Alloc >::insert(), std::map< \_Key, \_Tp, \_Compare, \_Alloc >::key\_comp(), std::map< \_Key, \_Tp, \_Compare, \_Alloc >::lower\_bound(), and std::piecewise\_construct.

```
4.625.3.35  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp> >> reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc
            >::rbegin ( ) [inline], [noexcept]
```

Returns a read/write reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 356 of file stl\_map.h.

```
4.625.3.36  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp> >> const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc
            >::rbegin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 365 of file stl\_map.h.

```
4.625.3.37  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp> >> reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rend (
            ) [inline], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 374 of file stl\_map.h.

```
4.625.3.38  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp> >> const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc
            >::rend ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 383 of file stl\_map.h.

```
4.625.3.39  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp> >> size_type std::map< _Key, _Tp, _Compare, _Alloc >::size ( ) const
            [inline], [noexcept]
```

Returns the size of the map.

Definition at line 434 of file stl\_map.h.

```
4.625.3.40 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =  
std::allocator<std::pair<const _Key, _Tp> >> void std::map<_Key, _Tp, _Compare, _Alloc >::swap ( map<  
_Key, _Tp, _Compare, _Alloc > &__x ) [inline]
```

Swaps data with another map.



## Parameters

<code>__x</code>	A map of the same element and allocator types.
------------------	--

This exchanges the elements between two maps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

Definition at line 778 of file `stl_map.h`.

Referenced by `std::map<_Key, _Tp, _Compare, _Alloc >::operator=()`, and `std::swap()`.

```
4.625.3.41  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp> >> iterator std::map<_Key, _Tp, _Compare, _Alloc >::upper_bound (
            const key_type & __x ) [inline]
```

Finds the end of a subsequence matching given key.

## Parameters

<code>__x</code>	Key of (key, value) pair to be located.
------------------	---

## Returns

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 888 of file `stl_map.h`.

```
4.625.3.42  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::map<_Key, _Tp, _Compare, _Alloc
            >::upper_bound ( const key_type & __x ) const [inline]
```

Finds the end of a subsequence matching given key.

## Parameters

<code>__x</code>	Key of (key, value) pair to be located.
------------------	---

## Returns

Read-only (constant) iterator pointing to first iterator greater than key, or `end()`.

Definition at line 898 of file `stl_map.h`.

```
4.625.3.43  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp> >> value_compare std::map<_Key, _Tp, _Compare, _Alloc
            >::value_comp ( ) const [inline]
```

Returns a value comparison object, built from the key comparison object out of which the map was constructed.

Definition at line 805 of file `stl_map.h`.

The documentation for this class was generated from the following file:

- [stl\\_map.h](#)

## 4.626 `std::mask_array<_Tp >` Class Template Reference

## Public Types

- typedef **\_Tp value\_type**

## Public Member Functions

- **mask\_array** (const **mask\_array** &)
- void **operator%=(const valarray< \_Tp > &)** const
- template<class \_Dom > void **operator%=(const \_Expr< \_Dom, \_Tp > &)** const
- void **operator&=(const valarray< \_Tp > &)** const
- template<class \_Dom > void **operator&=(const \_Expr< \_Dom, \_Tp > &)** const
- void **operator\*=(const valarray< \_Tp > &)** const
- template<class \_Dom > void **operator\*=(const \_Expr< \_Dom, \_Tp > &)** const
- void **operator+=(const valarray< \_Tp > &)** const
- template<class \_Dom > void **operator+=(const \_Expr< \_Dom, \_Tp > &)** const
- void **operator-=(const valarray< \_Tp > &)** const
- template<class \_Dom > void **operator-=(const \_Expr< \_Dom, \_Tp > &)** const
- void **operator/=(const valarray< \_Tp > &)** const
- template<class \_Dom > void **operator/=(const \_Expr< \_Dom, \_Tp > &)** const
- void **operator<<=(const valarray< \_Tp > &)** const
- template<class \_Dom > void **operator<<=(const \_Expr< \_Dom, \_Tp > &)** const
- **mask\_array** & **operator=(const mask\_array &)**
- void **operator=(const valarray< \_Tp > &)** const
- void **operator=(const \_Tp &)** const
- template<class \_Dom > void **operator=(const \_Expr< \_Dom, \_Tp > &)** const
- template<class \_Ex > void **operator=(const \_Expr< \_Ex, \_Tp > &\_\_e)** const
- void **operator>>=(const valarray< \_Tp > &)** const
- template<class \_Dom > void **operator>>=(const \_Expr< \_Dom, \_Tp > &)** const
- void **operator^=(const valarray< \_Tp > &)** const
- template<class \_Dom > void **operator^=(const \_Expr< \_Dom, \_Tp > &)** const
- void **operator|=(const valarray< \_Tp > &)** const
- template<class \_Dom > void **operator|=(const \_Expr< \_Dom, \_Tp > &)** const

## Friends

- class **valarray< \_Tp >**

## 4.626.1 Detailed Description

template<class \_Tp>class std::mask\_array< \_Tp >

Reference to selected subset of an array.

A **mask\_array** is a reference to the actual elements of an array specified by a bitmask in the form of an array of **bool**. The way to get a **mask\_array** is to call **operator[]**(**valarray<bool>**) on a **valarray**. The returned **mask\_array** then permits carrying operations out on the referenced subset of elements in the original **valarray**.

For example, if a **mask\_array** is obtained using the array (false, true, false, true) as an argument, the mask array has two elements referring to **array[1]** and **array[3]** in the underlying array.

## Parameters

<i>Tp</i>	Element type.
-----------	---------------

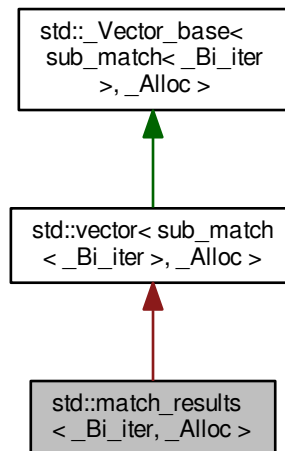
Definition at line 62 of file mask\_array.h.

The documentation for this class was generated from the following file:

- [mask\\_array.h](#)

#### 4.627 `std::match_results<_Bi_iter, _Alloc>` Class Template Reference

Inheritance diagram for `std::match_results<_Bi_iter, _Alloc>`:



## Public Member Functions

- bool `ready` () const

## Private Types

- typedef `_Alloc_traits::const_pointer` **const\_pointer**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::pointer` **pointer**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**

## Private Member Functions

- pointer `_M_allocate` (size\_t \_\_n)
- pointer `_M_allocate_and_copy` (size\_type \_\_n, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last)

- void **\_M\_assign\_aux** (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- void **\_M\_assign\_aux** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- void **\_M\_assign\_dispatch** (\_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- void **\_M\_assign\_dispatch** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- size\_type **\_M\_check\_len** (size\_type \_\_n, const char \*\_\_s) const
- void **\_M\_deallocate** (pointer \_\_p, size\_t \_\_n)
- void **\_M\_default\_append** (size\_type \_\_n)
- void **\_M\_default\_initialize** (size\_type \_\_n)
- void **\_M\_emplace\_back\_aux** (\_Args &&...\_\_args)
- void **\_M\_erase\_at\_end** (pointer \_\_pos)
- void **\_M\_fill\_assign** (size\_type \_\_n, const [value\\_type](#) &\_\_val)
- void **\_M\_fill\_initialize** (size\_type \_\_n, const [value\\_type](#) &\_\_value)
- void **\_M\_fill\_insert** (iterator \_\_pos, size\_type \_\_n, const [value\\_type](#) &\_\_x)
- \_Tp\_alloc\_type & **\_M\_get\_Tp\_allocator** () noexcept
- const \_Tp\_alloc\_type & **\_M\_get\_Tp\_allocator** () const noexcept
- void **\_M\_initialize\_dispatch** (\_Integer \_\_n, \_Integer \_\_value, \_\_true\_type)
- void **\_M\_initialize\_dispatch** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- void **\_M\_insert\_aux** (iterator \_\_position, \_Args &&...\_\_args)
- void **\_M\_insert\_dispatch** (iterator \_\_pos, \_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- void **\_M\_insert\_dispatch** (iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- void **\_M\_range\_check** (size\_type \_\_n) const
- void **\_M\_range\_initialize** (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- void **\_M\_range\_initialize** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- void **\_M\_range\_insert** (iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- void **\_M\_range\_insert** (iterator \_\_pos, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- bool **\_M\_shrink\_to\_fit** ()
- void **assign** (size\_type \_\_n, const [value\\_type](#) &\_\_val)
- void **assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void **assign** (initializer\_list< [value\\_type](#) > \_\_l)
- [reference](#) at (size\_type \_\_n)
- [const\\_reference](#) at (size\_type \_\_n) const
- [reference](#) back ()
- [const\\_reference](#) back () const
- iterator **begin** () noexcept
- size\_type **capacity** () const noexcept
- void **clear** () noexcept
- [const\\_reverse\\_iterator](#) **crbegin** () const noexcept
- [const\\_reverse\\_iterator](#) **crend** () const noexcept
- [sub\\_match](#)<\_Bi\_iter> \* **data** () noexcept
- const [sub\\_match](#)<\_Bi\_iter> \* **data** () const noexcept
- iterator **emplace** (iterator \_\_position, \_Args &&...\_\_args)
- void **emplace\_back** (\_Args &&...\_\_args)
- iterator **end** () noexcept
- iterator **erase** (iterator \_\_position)
- iterator **erase** (iterator \_\_first, iterator \_\_last)
- [reference](#) front ()
- [const\\_reference](#) front () const
- iterator **insert** (iterator \_\_position, const [value\\_type](#) &\_\_x)
- iterator **insert** (iterator \_\_position, [value\\_type](#) &&\_\_x)
- void **insert** (iterator \_\_position, initializer\_list< [value\\_type](#) > \_\_l)

- void `insert` (iterator \_\_position, size\_type \_\_n, const `value_type` &\_\_x)
- void `insert` (iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- `reference operator[]` (size\_type \_\_n)
- `const_reference operator[]` (size\_type \_\_n) const
- void `pop_back` ()
- void `push_back` (const `value_type` &\_\_x)
- void `push_back` (`value_type` &&\_\_x)
- `reverse_iterator rbegin` () noexcept
- `const_reverse_iterator rbegin` () const noexcept
- `reverse_iterator rend` () noexcept
- `const_reverse_iterator rend` () const noexcept
- void `reserve` (size\_type \_\_n)
- void `resize` (size\_type \_\_new\_size)
- void `resize` (size\_type \_\_new\_size, const `value_type` &\_\_x)
- void `shrink_to_fit` ()
- void `swap` (`vector` &\_\_x) noexcept(\_Alloc\_traits::\_S\_nothrow\_swap())

#### Private Attributes

- `_Vector_impl _M_impl`

#### Friends

- class `__detail::SpecializedResults< _Bi_iter, _Alloc >`

#### 10.? Public Types

- typedef `_Alloc allocator_type`
- typedef `sub_match< _Bi_iter > value_type`
- typedef const `value_type` & `const_reference`
- typedef `const_reference reference`
- typedef `_Base_type::const_iterator` `const_iterator`
- typedef `const_iterator` `iterator`
- typedef `__iter_traits::difference_type` `difference_type`
- typedef `__iter_traits::value_type` `char_type`
- typedef `allocator_traits< _Alloc >::size_type` `size_type`
- typedef `std::basic_string< char_type >` `string_type`

#### 28.10.1 Construction, Copying, and Destruction

- `match_results` (const `_Alloc` &\_\_a= `_Alloc`())
- `match_results` (const `match_results` &\_\_rhs)
- `match_results` (`match_results` &&\_\_rhs) noexcept
- `match_results` & `operator=` (const `match_results` &\_\_rhs)
- `match_results` & `operator=` (`match_results` &&\_\_rhs)
- `~match_results` ()

## 28.10.2 Size

- size\_type [size](#) () const
- size\_type [max\\_size](#) () const
- bool [empty](#) () const

## 10.3 Element Access

- difference\_type [length](#) (size\_type \_\_sub=0) const
- difference\_type [position](#) (size\_type \_\_sub=0) const
- string\_type [str](#) (size\_type \_\_sub=0) const
- [const\\_reference operator\[\]](#) (size\_type \_\_sub) const
- [const\\_reference prefix](#) () const
- [const\\_reference suffix](#) () const
- const\_iterator [begin](#) () const
- const\_iterator [cbegin](#) () const
- const\_iterator [end](#) () const
- const\_iterator [cend](#) () const

## 10.4 Formatting

These functions perform formatted substitution of the matched character sequences into their target. The format specifiers and escape sequences accepted by these functions are determined by their `flags` parameter as documented above.

- `template<typename _Out_iter> _Out_iter format (_Out_iter __out, const char_type *__fmt_first, const char_type *__fmt_last, match_flag_type __flags=regex\_constants::format\_default) const`
- `template<typename _Out_iter, typename _St, typename _Sa> _Out_iter format (_Out_iter __out, const basic\_string<char_type, _St, _Sa> &__fmt, match_flag_type __flags=regex\_constants::format\_default) const`
- `template<typename _Out_iter, typename _St, typename _Sa> basic\_string<char_type, _St, _Sa> format (const basic\_string<char_type, _St, _Sa> &__fmt, match_flag_type __flags=regex\_constants::format\_default) const`
- `string\_type format (const char_type *__fmt, match_flag_type __flags=regex\_constants::format\_default) const`

## 10.5 Allocator

- allocator\_type [get\\_allocator](#) () const

## 10.6 Swap

- void [swap](#) ([match\\_results](#) &\_\_that)

## 4.627.1 Detailed Description

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> class std::match_results<_Bi_iter, _Alloc>
```

The results of a match or search operation.

A collection of character sequences representing the result of a regular expression match. Storage for the collection is allocated and freed as necessary by the member functions of class template `match_results`.

This class satisfies the Sequence requirements, with the exception that only the operations defined for a const-qualified Sequence are supported.

The `sub_match` object stored at index 0 represents sub-expression 0, i.e. the whole match. In this case the `sub_match` member `matched` is always true. The `sub_match` object stored at index `n` denotes what matched the marked sub-expression `n` within the matched expression. If the sub-expression `n` participated in a regular expression match then the `sub_match` member `matched` evaluates to true, and members `first` and `second` denote the range of characters `[first, second)` which formed that match. Otherwise `matched` is false, and members `first` and `second` point to the end of the sequence that was searched.

Definition at line 1429 of file `regex.h`.

#### 4.627.2 Constructor & Destructor Documentation

**4.627.2.1** `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> std::match_results<_Bi_iter, _Alloc>::match_results ( const _Alloc & __a = _Alloc() ) [inline], [explicit]`

Constructs a default `match_results` container.

##### Postcondition

`size()` returns 0 and `str()` returns an empty string.

Definition at line 1478 of file `regex.h`.

Referenced by `std::match_results<_FwdIterT, _Alloc>::operator=()`.

**4.627.2.2** `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> std::match_results<_Bi_iter, _Alloc>::match_results ( const match_results<_Bi_iter, _Alloc> & __rhs ) [inline]`

Copy constructs a `match_results`.

Definition at line 1485 of file `regex.h`.

**4.627.2.3** `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> std::match_results<_Bi_iter, _Alloc>::match_results ( match_results<_Bi_iter, _Alloc> && __rhs ) [inline], [noexcept]`

Move constructs a `match_results`.

Definition at line 1492 of file `regex.h`.

**4.627.2.4** `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> std::match_results<_Bi_iter, _Alloc>::~~match_results ( ) [inline]`

Destroys a `match_results` object.

Definition at line 1519 of file `regex.h`.

#### 4.627.3 Member Function Documentation

**4.627.3.1** `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> const_iterator std::match_results<_Bi_iter, _Alloc>::begin ( ) const [inline]`

Gets an iterator to the start of the `sub_match` collection.

Definition at line 1676 of file `regex.h`.

Referenced by `std::operator==( )`.

4.627.3.2 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> const_iterator  
std::match_results<_Bi_iter, _Alloc>::cbegin ( ) const [inline]`

Gets an iterator to the start of the sub\_match collection.

Definition at line 1683 of file regex.h.

4.627.3.3 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> const_iterator  
std::match_results<_Bi_iter, _Alloc>::cend ( ) const [inline]`

Gets an iterator to one-past-the-end of the collection.

Definition at line 1697 of file regex.h.

4.627.3.4 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> bool std::match_results<  
_Bi_iter, _Alloc>::empty ( ) const [inline]`

Indicates if the match\_results contains no results.

Return values

<i>true</i>	The match_results object is empty.
<i>false</i>	The match_results object is not empty.

Definition at line 1563 of file regex.h.

Referenced by std::match\_results<\_FwdIterT, \_Alloc>::end(), std::operator==( ), std::match\_results<\_FwdIterT, \_Alloc>::prefix(), and std::match\_results<\_FwdIterT, \_Alloc>::suffix().

4.627.3.5 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> const_iterator  
std::match_results<_Bi_iter, _Alloc>::end ( ) const [inline]`

Gets an iterator to one-past-the-end of the collection.

Definition at line 1690 of file regex.h.

Referenced by std::match\_results<\_FwdIterT, \_Alloc>::cend(), and std::operator==( ).

4.627.3.6 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> template<typename _Out_iter >  
_Out_iter std::match_results<_Bi_iter, _Alloc>::format ( _Out_iter __out, const char_type * __fmt_first, const  
char_type * __fmt_last, match_flag_type __flags = regex_constants::format_default ) const [inline]`

Precondition

ready() == true

**Todo** Implement this function.

Definition at line 1718 of file regex.h.

Referenced by std::match\_results<\_FwdIterT, \_Alloc>::format().

4.627.3.7 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> template<typename _Out_iter ,  
typename _St, typename _Sa > _Out_iter std::match_results<_Bi_iter, _Alloc>::format ( _Out_iter __out, const  
basic_string< char_type, _St, _Sa > & __fmt, match_flag_type __flags = regex_constants::format_default )  
const [inline]`

Precondition

ready() == true

Definition at line 1728 of file regex.h.



```
4.627.3.8  template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> template<typename
            _Out_iter, typename _St, typename _Sa > basic_string<char_type, _St, _Sa> std::match_results<
            _Bi_iter, _Alloc >::format ( const basic_string< char_type, _St, _Sa > & __fmt, match_flag_type __flags =
            regex_constants::format_default ) const    [inline]
```

Precondition

ready() == true

Definition at line 1740 of file regex.h.

```
4.627.3.9  template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> string_type
            std::match_results< _Bi_iter, _Alloc >::format ( const char_type * __fmt, match_flag_type __flags =
            regex_constants::format_default ) const    [inline]
```

Precondition

ready() == true

Definition at line 1752 of file regex.h.

```
4.627.3.10 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> allocator_type
            std::match_results< _Bi_iter, _Alloc >::get_allocator ( ) const    [inline]
```

Gets a copy of the allocator.

Definition at line 1773 of file regex.h.

```
4.627.3.11 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> difference_type
            std::match_results< _Bi_iter, _Alloc >::length ( size_type __sub = 0 ) const    [inline]
```

Gets the length of the indicated submatch.

Parameters

<code>__sub</code>	indicates the submatch.
--------------------	-------------------------

Precondition

ready() == true

This function returns the length of the indicated submatch, or the length of the entire match if `__sub` is zero (the default).

Definition at line 1582 of file regex.h.

```
4.627.3.12 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> size_type
            std::match_results< _Bi_iter, _Alloc >::max_size ( ) const    [inline]
```

Gets the number of matches and submatches.

The number of matches for a given regular expression will be either 0 if there was no match or `mark_count() + 1` if a match was successful. Some matches may be empty.

Returns

the number of matches found.

Definition at line 1554 of file regex.h.

```
4.627.3.13 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> match_results&
std::match_results<_Bi_iter, _Alloc >::operator= ( const match_results<_Bi_iter, _Alloc > & __rhs )
[inline]
```

Assigns rhs to \*this.

Definition at line 1500 of file regex.h.

```
4.627.3.14 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> match_results&
std::match_results<_Bi_iter, _Alloc >::operator= ( match_results<_Bi_iter, _Alloc > && __rhs )
[inline]
```

Move-assigns rhs to \*this.

Definition at line 1510 of file regex.h.

```
4.627.3.15 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> const_reference
std::match_results<_Bi_iter, _Alloc >::operator[] ( size_type __sub ) const [inline]
```

Gets a sub\_match reference for the match or submatch.

Parameters

__sub	indicates the submatch.
-------	-------------------------

Precondition

ready() == true

This function gets a reference to the indicated submatch, or the entire match if \_\_sub is zero.

If \_\_sub >= size() then this function returns a sub\_match with a special value indicating no submatch.

Definition at line 1630 of file regex.h.

```
4.627.3.16 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> difference_type
std::match_results<_Bi_iter, _Alloc >::position ( size_type __sub = 0 ) const [inline]
```

Gets the offset of the beginning of the indicated submatch.

Parameters

__sub	indicates the submatch.
-------	-------------------------

Precondition

ready() == true

This function returns the offset from the beginning of the target sequence to the beginning of the submatch, unless the value of \_\_sub is zero (the default), in which case this function returns the offset from the beginning of the target sequence to the beginning of the match.

Returns -1 if \_\_sub is out of range.

Definition at line 1599 of file regex.h.

```
4.627.3.17 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> const_reference
std::match_results<_Bi_iter, _Alloc >::prefix ( ) const [inline]
```

Gets a sub\_match representing the match prefix.

**Precondition**

`ready() == true`

This function gets a reference to a `sub_match` object representing the part of the target range between the start of the target range and the start of the match.

Definition at line 1647 of file `regex.h`.

Referenced by `std::operator==()`, and `std::match_results<_FwdIterT, _Alloc>::position()`.

**4.627.3.18** `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> bool std::match_results<_Bi_iter, _Alloc>::ready( ) const [inline]`

Indicates if the `match_results` is ready.

**Return values**

<i>true</i>	The object has a fully-established result state.
<i>false</i>	The object is not ready.

Definition at line 1530 of file `regex.h`.

Referenced by `std::operator==()`, `std::match_results<_FwdIterT, _Alloc>::operator[]()`, `std::match_results<_FwdIterT, _Alloc>::prefix()`, and `std::match_results<_FwdIterT, _Alloc>::suffix()`.

**4.627.3.19** `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> size_type std::match_results<_Bi_iter, _Alloc>::size( ) const [inline]`

Gets the number of matches and submatches.

The number of matches for a given regular expression will be either 0 if there was no match or `mark_count() + 1` if a match was successful. Some matches may be empty.

**Returns**

the number of matches found.

Definition at line 1547 of file `regex.h`.

Referenced by `std::match_results<_FwdIterT, _Alloc>::empty()`, `std::operator==()`, `std::match_results<_FwdIterT, _Alloc>::operator[]()`, and `std::match_results<_FwdIterT, _Alloc>::position()`.

**4.627.3.20** `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> string_type std::match_results<_Bi_iter, _Alloc>::str( size_type __sub = 0 ) const [inline]`

Gets the match or submatch converted to a string type.

**Parameters**

<code>__sub</code>	indicates the submatch.
--------------------	-------------------------

**Precondition**

`ready() == true`

This function gets the submatch (or match, if `__sub` is zero) extracted from the target range and converted to the associated string type.

Definition at line 1615 of file `regex.h`.

4.627.3.21 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> const_reference  
std::match_results<_Bi_iter, _Alloc>::suffix ( ) const [inline]`

Gets a sub\_match representing the match suffix.

#### Precondition

`ready() == true`

This function gets a reference to a sub\_match object representing the part of the target range between the end of the match and the end of the target range.

Definition at line 1664 of file regex.h.

Referenced by std::operator==().

4.627.3.22 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> void std::match_results<  
_Bi_iter, _Alloc>::swap ( match_results<_Bi_iter, _Alloc> &__that ) [inline]`

Swaps the contents of two match\_results.

Definition at line 1787 of file regex.h.

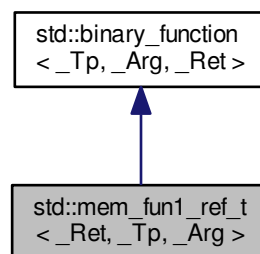
Referenced by std::swap().

The documentation for this class was generated from the following file:

- [regex.h](#)

## 4.628 std::mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg > Class Template Reference

Inheritance diagram for std::mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >:



#### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Ret` [result\\_type](#)
- typedef `_Arg` [second\\_argument\\_type](#)

## Public Member Functions

- **mem\_fun1\_ref\_t** (**\_Ret**(**\_Tp**::\*\_\_pf)(**\_Arg**))
- **\_Ret operator()** (**\_Tp** &\_\_r, **\_Arg** \_\_x) const

### 4.628.1 Detailed Description

template<typename **\_Ret**, typename **\_Tp**, typename **\_Arg**>class std::mem\_fun1\_ref\_t< **\_Ret**, **\_Tp**, **\_Arg** >

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

Definition at line 650 of file stl\_function.h.

### 4.628.2 Member Typedef Documentation

4.628.2.1 typedef **\_Tp** std::binary\_function< **\_Tp**, **\_Arg**, **\_Ret** >::first\_argument\_type [inherited]

first\_argument\_type is the type of the first argument

Definition at line 117 of file stl\_function.h.

4.628.2.2 typedef **\_Ret** std::binary\_function< **\_Tp**, **\_Arg**, **\_Ret** >::result\_type [inherited]

result\_type is the return type

Definition at line 123 of file stl\_function.h.

4.628.2.3 typedef **\_Arg** std::binary\_function< **\_Tp**, **\_Arg**, **\_Ret** >::second\_argument\_type [inherited]

second\_argument\_type is the type of the second argument

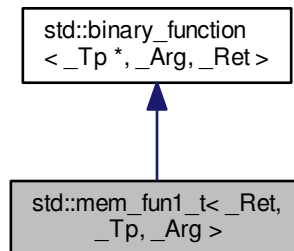
Definition at line 120 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 4.629 std::mem\_fun1\_t&lt; \_Ret, \_Tp, \_Arg &gt; Class Template Reference

Inheritance diagram for std::mem\_fun1\_t< \_Ret, \_Tp, \_Arg >:



## Public Types

- typedef \_Tp \* [first\\_argument\\_type](#)
- typedef \_Ret [result\\_type](#)
- typedef \_Arg [second\\_argument\\_type](#)

## Public Member Functions

- **mem\_fun1\_t** (\_Ret(\_Tp::\* \_\_pf)(\_Arg))
- \_Ret **operator()** (\_Tp \* \_\_p, \_Arg \_\_x) const

## 4.629.1 Detailed Description

```
template<typename _Ret, typename _Tp, typename _Arg>class std::mem_fun1_t< _Ret, _Tp, _Arg >
```

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

Definition at line 614 of file stl\_function.h.

## 4.629.2 Member Typedef Documentation

4.629.2.1 typedef \_Tp \* **std::binary\_function< \_Tp \*, \_Arg, \_Ret >::first\_argument\_type** [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl\_function.h.

4.629.2.2 typedef \_Ret **std::binary\_function< \_Tp \*, \_Arg, \_Ret >::result\_type** [inherited]

`result_type` is the return type

Definition at line 123 of file stl\_function.h.

#### 4.629.2.3 `typedef _Arg std::binary_function<_Tp*, _Arg, _Ret>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

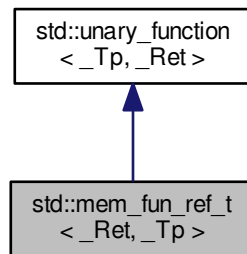
Definition at line 120 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

### 4.630 `std::mem_fun_ref_t<_Ret, _Tp>` Class Template Reference

Inheritance diagram for `std::mem_fun_ref_t<_Ret, _Tp>`:



#### Public Types

- `typedef _Tp` [argument\\_type](#)
- `typedef _Ret` [result\\_type](#)

#### Public Member Functions

- `mem_fun_ref_t(_Ret(_Tp::*__pf)())`
- `_Ret operator()(_Tp &__r) const`

#### 4.630.1 Detailed Description

`template<typename _Ret, typename _Tp> class std::mem_fun_ref_t<_Ret, _Tp>`

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

Definition at line 578 of file `stl_function.h`.

#### 4.630.2 Member Typedef Documentation

4.630.2.1 `typedef _Tp std::unary_function<_Tp,_Ret>::argument_type` [inherited]

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

4.630.2.2 `typedef _Ret std::unary_function<_Tp,_Ret>::result_type` [inherited]

`result_type` is the return type

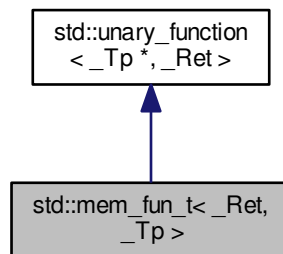
Definition at line 107 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 4.631 `std::mem_fun_t<_Ret,_Tp>` Class Template Reference

Inheritance diagram for `std::mem_fun_t<_Ret,_Tp>`:



### Public Types

- `typedef _Tp *` [argument\\_type](#)
- `typedef _Ret` [result\\_type](#)

### Public Member Functions

- `mem_fun_t` (`_Ret` (`_Tp::*` `__pf`))()
- `_Ret operator()` (`_Tp *` `__p`) `const`

### 4.631.1 Detailed Description

`template<typename _Ret, typename _Tp> class std::mem_fun_t<_Ret,_Tp>`

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

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



#### 4.631.2 Member Typedef Documentation

##### 4.631.2.1 `typedef _Tp * std::unary_function< _Tp *, _Ret >::argument_type` [inherited]

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

##### 4.631.2.2 `typedef _Ret std::unary_function< _Tp *, _Ret >::result_type` [inherited]

`result_type` is the return type

Definition at line 107 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

#### 4.632 `std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >` > Class Template Reference

##### Public Types

- `typedef _UIntType` [result\\_type](#)

##### Public Member Functions

- `mersenne_twister_engine` ([result\\_type](#) \_\_sd=default\_seed)
- `template<typename _Sseq , typename = typename std::enable_if<!std::is_same<_Sseq, mersenne_twister_engine>::value> ::type>`  
`mersenne_twister_engine` (\_Sseq &\_\_q)
- `void` [discard](#) (unsigned long long \_\_z)
- `result_type` [operator\(\)](#) ()
- `void` [seed](#) ([result\\_type](#) \_\_sd=default\_seed)
- `template<typename _Sseq > std::enable_if< std::is_class< _Sseq >::value >::type` [seed](#) (\_Sseq &\_\_q)

##### Static Public Member Functions

- `static constexpr` [result\\_type](#) [max](#) ()
- `static constexpr` [result\\_type](#) [min](#) ()

##### Static Public Attributes

- `static constexpr` [result\\_type](#) [default\\_seed](#)
- `static constexpr` [result\\_type](#) [initialization\\_multiplier](#)
- `static constexpr` `size_t` [mask\\_bits](#)
- `static constexpr` `size_t` [shift\\_size](#)
- `static constexpr` `size_t` [state\\_size](#)
- `static constexpr` [result\\_type](#) [tempering\\_b](#)
- `static constexpr` [result\\_type](#) [tempering\\_c](#)
- `static constexpr` [result\\_type](#) [tempering\\_d](#)
- `static constexpr` `size_t` [tempering\\_l](#)
- `static constexpr` `size_t` [tempering\\_s](#)

- static constexpr size\_t **tempering\_t**
- static constexpr size\_t **tempering\_u**
- static constexpr size\_t **word\_size**
- static constexpr [result\\_type](#) **xor\_mask**

## Friends

- template<typename \_UIntType1, size\_t \_\_w1, size\_t \_\_n1, size\_t \_\_m1, size\_t \_\_r1, \_UIntType1 \_\_a1, size\_t \_\_u1, \_UIntType1 \_\_d1, size\_t \_\_s1, \_UIntType1 \_\_b1, size\_t \_\_t1, \_UIntType1 \_\_c1, size\_t \_\_l1, \_UIntType1 \_\_f1, typename \_CharT, typename \_Traits> std::basic\_ostream< \_CharT, \_Traits> & [operator<<](#) (std::basic\_ostream< \_CharT, \_Traits> &\_\_os, const [std::mersenne\\_twister\\_engine](#)< \_UIntType1, \_\_w1, \_\_n1, \_\_m1, \_\_r1, \_\_a1, \_\_u1, \_\_d1, \_\_s1, \_\_b1, \_\_t1, \_\_c1, \_\_l1, \_\_f1> &\_\_x)
- bool [operator==](#) (const [mersenne\\_twister\\_engine](#) &\_\_lhs, const [mersenne\\_twister\\_engine](#) &\_\_rhs)
- template<typename \_UIntType1, size\_t \_\_w1, size\_t \_\_n1, size\_t \_\_m1, size\_t \_\_r1, \_UIntType1 \_\_a1, size\_t \_\_u1, \_UIntType1 \_\_d1, size\_t \_\_s1, \_UIntType1 \_\_b1, size\_t \_\_t1, \_UIntType1 \_\_c1, size\_t \_\_l1, \_UIntType1 \_\_f1, typename \_CharT, typename \_Traits> std::basic\_istream< \_CharT, \_Traits> & [operator>>](#) (std::basic\_istream< \_CharT, \_Traits> &\_\_is, [std::mersenne\\_twister\\_engine](#)< \_UIntType1, \_\_w1, \_\_n1, \_\_m1, \_\_r1, \_\_a1, \_\_u1, \_\_d1, \_\_s1, \_\_b1, \_\_t1, \_\_c1, \_\_l1, \_\_f1> &\_\_x)

## 4.632.1 Detailed Description

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s,
        _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>class std::mersenne_twister_engine< _UIntType, __w, __n, __m,
        __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >
```

A generalized feedback shift register discrete random number generator.

This algorithm avoids multiplication and division and is designed to be friendly to a pipelined architecture. If the parameters are chosen correctly, this generator will produce numbers with a very long period and fairly good apparent entropy, although still not cryptographically strong.

The best way to use this generator is with the predefined mt19937 class.

This algorithm was originally invented by Makoto Matsumoto and Takuji Nishimura.

## Template Parameters

<code>__w</code>	Word size, the number of bits in each element of the state vector.
<code>__n</code>	The degree of recursion.
<code>__m</code>	The period parameter.
<code>__r</code>	The separation point bit index.
<code>__a</code>	The last row of the twist matrix.
<code>__u</code>	The first right-shift tempering matrix parameter.
<code>__d</code>	The first right-shift tempering matrix mask.
<code>__s</code>	The first left-shift tempering matrix parameter.
<code>__b</code>	The first left-shift tempering matrix mask.
<code>__t</code>	The second left-shift tempering matrix parameter.
<code>__c</code>	The second left-shift tempering matrix mask.
<code>__l</code>	The second right-shift tempering matrix parameter.

<code>__f</code>	Initialization multiplier.
------------------	----------------------------

Definition at line 449 of file random.h.

#### 4.632.2 Member Typedef Documentation

4.632.2.1 `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> typedef _UIntType std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::result_type`

The type of the generated random value.

Definition at line 452 of file random.h.

#### 4.632.3 Constructor & Destructor Documentation

4.632.3.1 `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> template<typename _Sseq, typename = typename std::enable_if<!std::is_same<_Sseq, mersenne_twister_engine>::value>::type> std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::mersenne_twister_engine ( _Sseq & __q ) [inline], [explicit]`

Constructs a mersenne\_twister\_engine random number generator engine seeded from the seed sequence `__q`.

Parameters

<code>__q</code>	the seed sequence.
------------------	--------------------

Definition at line 513 of file random.h.

#### 4.632.4 Member Function Documentation

4.632.4.1 `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> void std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::discard ( unsigned long long __z )`

Discard a sequence of random numbers.

4.632.4.2 `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> static constexpr result_type std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::max ( ) [inline], [static]`

Gets the largest possible value in the output range.

Definition at line 534 of file random.h.

4.632.4.3 `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> static constexpr result_type std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::min ( ) [inline], [static]`

Gets the smallest possible value in the output range.

Definition at line 527 of file random.h.

#### 4.632.5 Friends And Related Function Documentation

4.632.5.1 `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> template<typename _UIntType1, size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT, typename _Traits> std::basic_ostream<_CharT, _Traits>& operator<< ( std::basic_ostream<_CharT, _Traits> & __os, const std::mersenne_twister_engine<_UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1> & __x ) [friend]`

Inserts the current state of a % mersenne\_twister\_engine random number generator engine `__x` into the output stream `__os`.

##### Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A % mersenne_twister_engine random number generator engine.

##### Returns

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

4.632.5.2 `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> bool operator== ( const mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f> & __lhs, const mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f> & __rhs ) [friend]`

Compares two % mersenne\_twister\_engine random number generator objects of the same type for equality.

##### Parameters

<code>__lhs</code>	A % mersenne_twister_engine random number generator object.
<code>__rhs</code>	Another % mersenne_twister_engine random number generator object.

##### Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 559 of file random.h.

4.632.5.3 `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> template<typename _UIntType1, size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<_CharT, _Traits> & __is, std::mersenne_twister_engine<_UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1> & __x ) [friend]`

Extracts the current state of a % mersenne\_twister\_engine random number generator engine `__x` from the input stream `__is`.

## Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A % mersenne_twister_engine random number generator engine.

## Returns

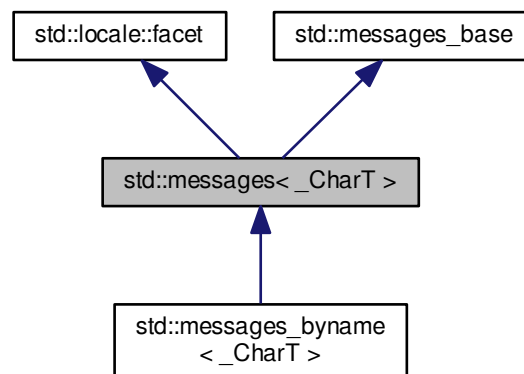
The input stream with the state of `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

4.633 `std::messages<_CharT>` Class Template Reference

Inheritance diagram for `std::messages<_CharT>`:



## Public Types

- typedef int **catalog**
- typedef `_CharT` [char\\_type](#)
- typedef [basic\\_string<\\_CharT>](#) [string\\_type](#)

## Public Member Functions

- [messages](#) (size\_t \_\_refs=0)
- [messages](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s, size\_t \_\_refs=0)
- void **close** (catalog \_\_c) const
- [string\\_type](#) **get** (catalog \_\_c, int \_\_set, int \_\_msgid, const [string\\_type](#) &\_\_s) const
- catalog **open** (const [basic\\_string](#)< char > &\_\_s, const [locale](#) &\_\_loc) const
- catalog **open** (const [basic\\_string](#)< char > &, const [locale](#) &, const char \*) const

## Static Public Attributes

- static [locale::id](#) id

## Protected Member Functions

- virtual [~messages](#) ()
- [string\\_type](#) [\\_M\\_convert\\_from\\_char](#) (char \*) const
- char \* [\\_M\\_convert\\_to\\_char](#) (const [string\\_type](#) &\_\_msg) const
- virtual void [do\\_close](#) (catalog) const
- virtual [string\\_type](#) [do\\_get](#) (catalog, int, int, const [string\\_type](#) &\_\_default) const
- template<> [string](#) [do\\_get](#) (catalog, int, int, const [string](#) &) const
- template<> [wstring](#) [do\\_get](#) (catalog, int, int, const [wstring](#) &) const
- virtual catalog [do\\_open](#) (const [basic\\_string](#)< char > &, const [locale](#) &) const

## Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \* \_\_s)

## Protected Attributes

- \_\_c\_locale [\\_M\\_c\\_locale\\_messages](#)
- const char \* [\\_M\\_name\\_messages](#)

## 4.633.1 Detailed Description

template<typename \_CharT>class std::messages<\_CharT>

Primary class template messages.

This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.

This library currently implements 3 versions of the message facet. The first version (gnu) is a wrapper around gettext, provided by libintl. The second version (ieee) is a wrapper around catgets. The final version (default) does no actual translation. These implementations are only provided for char and wchar\_t instantiations.

The messages template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the messages facet.

Definition at line 1695 of file locale\_facets\_nonio.h.

#### 4.633.2 Member Typedef Documentation

##### 4.633.2.1 `template<typename _CharT> typedef _CharT std::messages<_CharT>::char_type`

Public typedefs.

Definition at line 1701 of file `locale_facets_nonio.h`.

##### 4.633.2.2 `template<typename _CharT> typedef basic_string<_CharT> std::messages<_CharT>::string_type`

Public typedefs.

Definition at line 1702 of file `locale_facets_nonio.h`.

#### 4.633.3 Constructor & Destructor Documentation

##### 4.633.3.1 `template<typename _CharT> std::messages<_CharT>::messages ( size_t __refs = 0 ) [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

Definition at line 44 of file `messages_members.h`.

##### 4.633.3.2 `template<typename _CharT> std::messages<_CharT>::messages ( __c_locale __cloc, const char * __s, size_t __refs = 0 ) [explicit]`

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

Parameters

<code>__cloc</code>	The C locale.
<code>__s</code>	The name of a locale.
<code>__refs</code>	RefCount to pass to the base class.

Definition at line 50 of file `messages_members.h`.

##### 4.633.3.3 `template<typename _CharT> std::messages<_CharT>::~messages ( ) [protected],[virtual]`

Destructor.

Definition at line 79 of file `messages_members.h`.

#### 4.633.4 Member Data Documentation

##### 4.633.4.1 `template<typename _CharT> locale::id std::messages<_CharT>::id [static]`

Numpunct facet id.

Definition at line 1713 of file `locale_facets_nonio.h`.

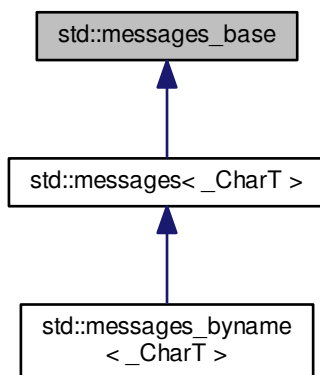
The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)

- [messages\\_members.h](#)

#### 4.634 std::messages\_base Struct Reference

Inheritance diagram for std::messages\_base:



##### Public Types

- typedef int **catalog**

##### 4.634.1 Detailed Description

Messages facet base class providing catalog typedef.

Definition at line 1668 of file `locale_facets_nonio.h`.

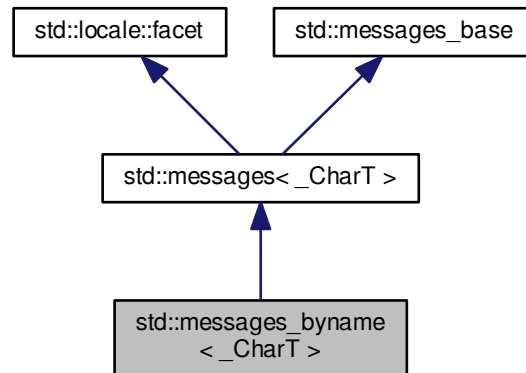
The documentation for this struct was generated from the following file:

- [locale\\_facets\\_nonio.h](#)



#### 4.635 `std::messages_byname<_CharT>` Class Template Reference

Inheritance diagram for `std::messages_byname<_CharT>`:



##### Public Types

- typedef int **catalog**
- typedef `_CharT` **char\_type**
- typedef `basic_string<_CharT>` **string\_type**

##### Public Member Functions

- **messages\_byname** (const char \*\_\_s, size\_t \_\_refs=0)
- void **close** (catalog \_\_c) const
- `string_type` **get** (catalog \_\_c, int \_\_set, int \_\_msgid, const `string_type` &\_\_s) const
- catalog **open** (const `basic_string<char>` &\_\_s, const `locale` &\_\_loc) const
- catalog **open** (const `basic_string<char>` &, const `locale` &, const char \*) const

##### Static Public Attributes

- static `locale::id` **id**

##### Protected Member Functions

- `string_type` **\_M\_convert\_from\_char** (char \*) const
- char \* **\_M\_convert\_to\_char** (const `string_type` &\_\_msg) const
- virtual void **do\_close** (catalog) const
- virtual `string_type` **do\_get** (catalog, int, int, const `string_type` &\_\_dfault) const
- template<> `string` **do\_get** (catalog, int, int, const `string` &) const
- template<> `wstring` **do\_get** (catalog, int, int, const `wstring` &) const
- virtual catalog **do\_open** (const `basic_string<char>` &, const `locale` &) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_messages**
- const char \* **\_M\_name\_messages**

## 4.635.1 Detailed Description

template<typename \_CharT>class std::messages\_byname<\_CharT>

class messages\_byname [22.2.7.2].

Definition at line 1879 of file locale\_facets\_nonio.h.

## 4.635.2 Member Data Documentation

4.635.2.1 template<typename \_CharT> locale::id std::messages<\_CharT>::id [static], [inherited]

Numpunct facet id.

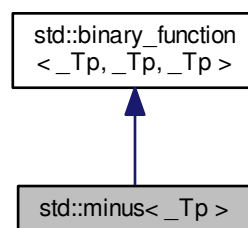
Definition at line 1713 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [messages\\_members.h](#)

## 4.636 std::minus&lt;\_Tp&gt; Struct Template Reference

Inheritance diagram for std::minus<\_Tp>:



## Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Tp` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

## Public Member Functions

- `_Tp` **operator()** (const `_Tp` &\_\_x, const `_Tp` &\_\_y) const

### 4.636.1 Detailed Description

template<typename `_Tp`>struct std::minus< `_Tp` >

One of the [math functors](#).

Definition at line 149 of file `stl_function.h`.

### 4.636.2 Member Typedef Documentation

4.636.2.1 typedef `_Tp` std::binary\_function< `_Tp`, `_Tp`, `_Tp` >::first\_argument\_type [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.636.2.2 typedef `_Tp` std::binary\_function< `_Tp`, `_Tp`, `_Tp` >::result\_type [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.636.2.3 typedef `_Tp` std::binary\_function< `_Tp`, `_Tp`, `_Tp` >::second\_argument\_type [inherited]

`second_argument_type` is the type of the second argument

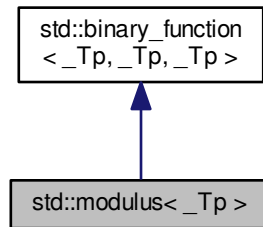
Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.637 std::modulus&lt;\_Tp&gt; Struct Template Reference

Inheritance diagram for std::modulus<\_Tp>:



## Public Types

- typedef \_Tp [first\\_argument\\_type](#)
- typedef \_Tp [result\\_type](#)
- typedef \_Tp [second\\_argument\\_type](#)

## Public Member Functions

- \_Tp **operator()** (const \_Tp &\_\_x, const \_Tp &\_\_y) const

## 4.637.1 Detailed Description

template<typename \_Tp>struct std::modulus<\_Tp>

One of the [math functors](#).

Definition at line 176 of file stl\_function.h.

## 4.637.2 Member Typedef Documentation

4.637.2.1 typedef \_Tp std::binary\_function<\_Tp, \_Tp, \_Tp>::first\_argument\_type [inherited]

first\_argument\_type is the type of the first argument

Definition at line 117 of file stl\_function.h.

4.637.2.2 typedef \_Tp std::binary\_function<\_Tp, \_Tp, \_Tp>::result\_type [inherited]

result\_type is the return type

Definition at line 123 of file stl\_function.h.

#### 4.637.2.3 `typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

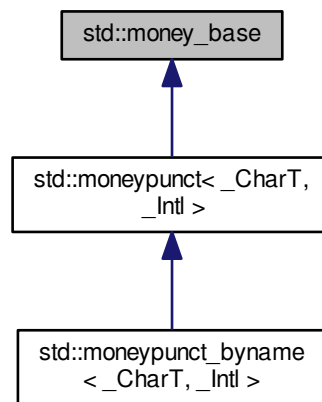
Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

#### 4.638 `std::money_base` Class Reference

Inheritance diagram for `std::money_base`:



##### Public Types

- enum { **\_S\_minus**, **\_S\_zero**, **\_S\_end** }
- enum **part** { **none**, **space**, **symbol**, **sign**, **value** }

##### Static Public Member Functions

- static pattern **\_S\_construct\_pattern** (char \_\_precedes, char \_\_space, char \_\_posn) throw ()

##### Static Public Attributes

- static const char \* **\_S\_atoms**
- static const pattern **\_S\_default\_pattern**

## 4.638.1 Detailed Description

Money format ordering data.

This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.

## See also

money\_punct::pos\_format() and money\_punct::neg\_format() for details of how these fields are interpreted.

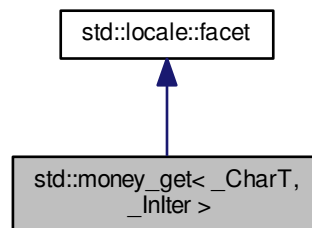
Definition at line 840 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 4.639 std::money\_get&lt; \_CharT, \_InIter &gt; Class Template Reference

Inheritance diagram for std::money\_get< \_CharT, \_InIter >:



## Public Types

- typedef `_CharT` [char\\_type](#)
- typedef `_InIter` [iter\\_type](#)
- typedef [basic\\_string< \\_CharT >](#) [string\\_type](#)

## Public Member Functions

- [money\\_get](#) (size\_t \_\_refs=0)
- [iter\\_type get](#) ([iter\\_type](#) \_\_s, [iter\\_type](#) \_\_end, bool \_\_intl, [ios\\_base](#) & \_\_io, [ios\\_base::iostate](#) & \_\_err, long double & \_\_units) const
- [iter\\_type get](#) ([iter\\_type](#) \_\_s, [iter\\_type](#) \_\_end, bool \_\_intl, [ios\\_base](#) & \_\_io, [ios\\_base::iostate](#) & \_\_err, [string\\_type](#) & \_\_digits) const

## Static Public Attributes

- static `locale::id` `id`

## Protected Member Functions

- virtual `~money_get()`
- `template<bool _Intl> iter_type M_extract(iter_type __s, iter_type __end, ios_base &__io, ios_base::iostate &__err, string &__digits) const`
- virtual `iter_type do_get(iter_type __s, iter_type __end, bool __intl, ios_base &__io, ios_base::iostate &__err, long double &__units) const`
- virtual `iter_type do_get(iter_type __s, iter_type __end, bool __intl, ios_base &__io, ios_base::iostate &__err, string_type &__digits) const`

## Static Protected Member Functions

- static `__c_locale _S_clone_c_locale(__c_locale &__cloc) throw()`
- static void `_S_create_c_locale(__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale(__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale()`
- static const char \* `_S_get_c_name()` throw()
- static `__c_locale _S_lc_ctype_c_locale(__c_locale __cloc, const char *__s)`

### 4.639.1 Detailed Description

`template<typename _CharT, typename _InIter> class std::money_get<_CharT, _InIter>`

Primary class template `money_get`.

This facet encapsulates the code to parse and return a monetary amount from a string.

The `money_get` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `money_get` facet.

Definition at line 1370 of file `locale_facets_nonio.h`.

### 4.639.2 Member Typedef Documentation

4.639.2.1 `template<typename _CharT, typename _InIter> typedef _CharT std::money_get<_CharT, _InIter>::char_type`

Public typedefs.

Definition at line 1376 of file `locale_facets_nonio.h`.

4.639.2.2 `template<typename _CharT, typename _InIter> typedef _InIter std::money_get<_CharT, _InIter>::iter_type`

Public typedefs.

Definition at line 1377 of file `locale_facets_nonio.h`.

4.639.2.3 `template<typename _CharT, typename _Inlter > typedef basic_string<_CharT> std::money_get< _CharT, _Inlter >::string_type`

Public typedefs.

Definition at line 1378 of file locale\_facets\_nonio.h.

#### 4.639.3 Constructor & Destructor Documentation

4.639.3.1 `template<typename _CharT, typename _Inlter > std::money_get< _CharT, _Inlter >::money_get ( size_t __refs = 0 ) [inline], [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

Definition at line 1392 of file locale\_facets\_nonio.h.

4.639.3.2 `template<typename _CharT, typename _Inlter > virtual std::money_get< _CharT, _Inlter >::~~money_get ( ) [inline], [protected], [virtual]`

Destructor.

Definition at line 1460 of file locale\_facets\_nonio.h.

#### 4.639.4 Member Function Documentation

4.639.4.1 `template<typename _CharT, typename _Inlter > virtual iter_type std::money_get< _CharT, _Inlter >::do_get ( iter_type __s, iter_type __end, bool __intl, ios_base & __io, ios_base::iostate & __err, long double & __units ) const [protected], [virtual]`

Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

See also

`get()` for details.

Referenced by `std::money_get< _CharT, _Inlter >::get()`.

4.639.4.2 `template<typename _CharT, typename _Inlter > virtual iter_type std::money_get< _CharT, _Inlter >::do_get ( iter_type __s, iter_type __end, bool __intl, ios_base & __io, ios_base::iostate & __err, string_type & __digits ) const [protected], [virtual]`

Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

See also

`get()` for details.



4.639.4.3 `template<typename _CharT, typename _InIter> iter_type std::money_get<_CharT, _InIter>::get ( iter_type __s, iter_type __end, bool __intl, ios_base & __io, ios_base::iostate & __err, long double & __units ) const [inline]`

Read and parse a monetary value.

This function reads characters from `__s`, interprets them as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and returns the result in `units` as an integral value `moneypunct::frac_digits() * the actual amount`. For example, the string \$10.01 in a US locale would store 1001 in `units`.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets `err=(err|io.failbit)`. If the stream is consumed before finishing parsing, sets `err=(err|io.failbit|io.eofbit)`. `units` is unchanged if parsing fails.

This function works by returning the result of `do_get()`.

#### Parameters

<code>__s</code>	Start of characters to parse.
<code>__end</code>	End of characters to parse.
<code>__intl</code>	Parameter to use <code>_facet&lt;moneypunct&lt;CharT,intl&gt;&gt;</code> .
<code>__io</code>	Source of facets and io state.
<code>__err</code>	Error field to set if parsing fails.
<code>__units</code>	Place to store result of parsing.

#### Returns

Iterator referencing first character beyond valid money amount.

Definition at line 1422 of file `locale_facets_nonio.h`.

References `std::money_get<_CharT, _InIter>::do_get()`.

4.639.4.4 `template<typename _CharT, typename _InIter> iter_type std::money_get<_CharT, _InIter>::get ( iter_type __s, iter_type __end, bool __intl, ios_base & __io, ios_base::iostate & __err, string_type & __digits ) const [inline]`

Read and parse a monetary value.

This function reads characters from `__s`, interprets them as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and returns the result in `digits`. For example, the string \$10.01 in a US locale would store 1001 in `digits`.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets `err=(err|io.failbit)`. If the stream is consumed before finishing parsing, sets `err=(err|io.failbit|io.eofbit)`.

This function works by returning the result of `do_get()`.

#### Parameters

<code>__s</code>	Start of characters to parse.
<code>__end</code>	End of characters to parse.
<code>__intl</code>	Parameter to use <code>_facet&lt;moneypunct&lt;CharT,intl&gt;&gt;</code> .

<code>__io</code>	Source of facets and io state.
<code>__err</code>	Error field to set if parsing fails.
<code>__digits</code>	Place to store result of parsing.

**Returns**

Iterator referencing first character beyond valid money amount.

Definition at line 1453 of file locale\_facets\_nonio.h.

References `std::money_get< _CharT, _InIter >::do_get()`.

**4.639.5 Member Data Documentation**

**4.639.5.1** `template<typename _CharT, typename _InIter> locale::id std::money_get< _CharT, _InIter >::id` `[static]`

Numpunct facet id.

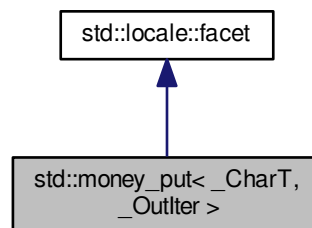
Definition at line 1382 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

**4.640 std::money\_put< \_CharT, \_OutIter > Class Template Reference**

Inheritance diagram for `std::money_put< _CharT, _OutIter >`:

**Public Types**

- typedef `_CharT` [char\\_type](#)
- typedef `_OutIter` [iter\\_type](#)
- typedef `basic_string< _CharT >` [string\\_type](#)

**Public Member Functions**

- [money\\_put](#) (`size_t __refs=0`)

- `iter_type put (iter_type __s, bool __intl, ios_base & __io, char_type __fill, long double __units) const`
- `iter_type put (iter_type __s, bool __intl, ios_base & __io, char_type __fill, const string_type & __digits) const`

#### Static Public Attributes

- static `locale::id id`

#### Protected Member Functions

- virtual `~money_put ()`
- `template<bool _Intl> iter_type _M_insert (iter_type __s, ios_base & __io, char_type __fill, const string_type & __digits) const`
- virtual `iter_type do_put (iter_type __s, bool __intl, ios_base & __io, char_type __fill, long double __units) const`
- virtual `iter_type do_put (iter_type __s, bool __intl, ios_base & __io, char_type __fill, const string_type & __digits) const`

#### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale & __cloc) throw ()`
- static void `_S_create_c_locale (__c_locale & __cloc, const char * __s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale & __cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_type_c_locale (__c_locale __cloc, const char * __s)`

#### 4.640.1 Detailed Description

`template<typename _CharT, typename _Outiter> class std::money_put< _CharT, _Outiter >`

Primary class template `money_put`.

This facet encapsulates the code to format and output a monetary amount.

The `money_put` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `money_put` facet.

Definition at line 1521 of file `locale_facets_nonio.h`.

#### 4.640.2 Member Typedef Documentation

4.640.2.1 `template<typename _CharT, typename _Outiter> typedef _CharT std::money_put< _CharT, _Outiter >::char_type`

Public typedefs.

Definition at line 1526 of file `locale_facets_nonio.h`.

4.640.2.2 `template<typename _CharT, typename _Outiter> typedef _Outiter std::money_put< _CharT, _Outiter >::iter_type`

Public typedefs.

Definition at line 1527 of file `locale_facets_nonio.h`.

4.640.2.3 `template<typename _CharT, typename _Outlter > typedef basic_string<_CharT> std::money_put<_CharT, _Outlter >::string_type`

Public typedefs.

Definition at line 1528 of file locale\_facets\_nonio.h.

#### 4.640.3 Constructor & Destructor Documentation

4.640.3.1 `template<typename _CharT, typename _Outlter > std::money_put<_CharT, _Outlter >::money_put ( size_t __refs = 0 ) [inline], [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

Definition at line 1542 of file locale\_facets\_nonio.h.

4.640.3.2 `template<typename _CharT, typename _Outlter > virtual std::money_put<_CharT, _Outlter >::~~money_put ( ) [inline], [protected], [virtual]`

Destructor.

Definition at line 1592 of file locale\_facets\_nonio.h.

#### 4.640.4 Member Function Documentation

4.640.4.1 `template<typename _CharT, typename _Outlter > virtual iter_type std::money_put<_CharT, _Outlter >::do_put ( iter_type __s, bool __intl, ios_base & __io, char_type __fill, long double __units ) const [protected], [virtual]`

Format and output a monetary value.

This function formats *units* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to *\_\_s*. For example, the value 1001 in a US locale would write \$10.01 to *\_\_s*.

This function is a hook for derived classes to change the value returned.

See also

`put()`.

Parameters

<code>__s</code>	The stream to write to.
<code>__intl</code>	Parameter to use <code>_facet&lt;moneypunct&lt;CharT,intl&gt; &gt;</code> .
<code>__io</code>	Source of facets and io state.
<code>__fill</code>	<code>char_type</code> to use for padding.
<code>__units</code>	Place to store result of parsing.

Returns

Iterator after writing.

Referenced by `std::money_put<_CharT, _Outlter >::put()`.

4.640.4.2 `template<typename _CharT, typename _Outiter> virtual iter_type std::money_put<_CharT, _Outiter>::do_put ( iter_type __s, bool __intl, ios_base & __io, char_type __fill, const string_type & __digits ) const [protected], [virtual]`

Format and output a monetary value.

This function formats *digits* as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and writes the resulting characters to `__s`. For example, the string `1001` in a US locale would write `$10.01` to `__s`.

This function is a hook for derived classes to change the value returned.

See also

`put()`.

#### Parameters

<code>__s</code>	The stream to write to.
<code>__intl</code>	Parameter to use <code>_facet&lt;moneypunct&lt;CharT,intl&gt;&gt;</code> .
<code>__io</code>	Source of facets and io state.
<code>__fill</code>	<code>char_type</code> to use for padding.
<code>__digits</code>	Place to store result of parsing.

#### Returns

Iterator after writing.

4.640.4.3 `template<typename _CharT, typename _Outiter> iter_type std::money_put<_CharT, _Outiter>::put ( iter_type __s, bool __intl, ios_base & __io, char_type __fill, long double __units ) const [inline]`

Format and output a monetary value.

This function formats *units* as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and writes the resulting characters to `__s`. For example, the value `1001` in a US locale would write `$10.01` to `__s`.

This function works by returning the result of `do_put()`.

#### Parameters

<code>__s</code>	The stream to write to.
<code>__intl</code>	Parameter to use <code>_facet&lt;moneypunct&lt;CharT,intl&gt;&gt;</code> .
<code>__io</code>	Source of facets and io state.
<code>__fill</code>	<code>char_type</code> to use for padding.
<code>__units</code>	Place to store result of parsing.

#### Returns

Iterator after writing.

Definition at line 1562 of file `locale_facets_nonio.h`.

References `std::money_put<_CharT, _Outiter>::do_put()`.

4.640.4.4 `template<typename _CharT, typename _Outiter> iter_type std::money_put<_CharT, _Outiter>::put ( iter_type __s, bool __intl, ios_base & __io, char_type __fill, const string_type & __digits ) const [inline]`

Format and output a monetary value.

This function formats *digits* as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and writes the resulting characters to `__s`. For example, the string `1001` in a US locale would write `$10.01` to `__s`.

This function works by returning the result of `do_put()`.

**Parameters**

<code>__s</code>	The stream to write to.
<code>__intl</code>	Parameter to use <code>_facet&lt;moneypunct&lt;CharT,intl&gt; &gt;</code> .
<code>__io</code>	Source of facets and io state.
<code>__fill</code>	<code>char_type</code> to use for padding.
<code>__digits</code>	Place to store result of parsing.

**Returns**

Iterator after writing.

Definition at line 1585 of file `locale_facets_nonio.h`.

References `std::money_put<_CharT, _Outlter >::do_put()`.

**4.640.5 Member Data Documentation**

**4.640.5.1** `template<typename _CharT, typename _Outlter> locale::id std::money_put<_CharT, _Outlter>::id`  
`[static]`

Numpunct facet id.

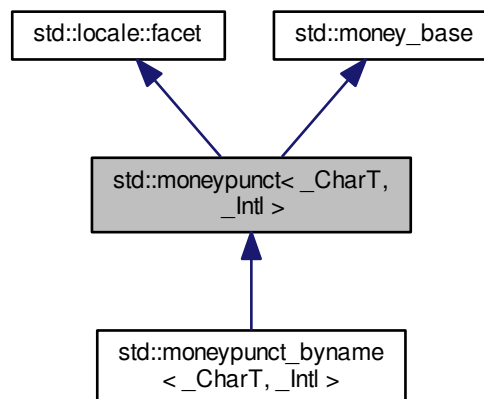
Definition at line 1532 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

**4.641 std::moneypunct<\_CharT, \_Intl> Class Template Reference**

Inheritance diagram for `std::moneypunct<_CharT, _Intl>`:



## Public Types

- enum { **\_S\_minus**, **\_S\_zero**, **\_S\_end** }
- typedef \_\_moneypunct\_cache<\_CharT, \_Intl> **\_\_cache\_type**
- enum **part** { **none**, **space**, **symbol**, **sign**, **value** }
- typedef \_CharT **char\_type**
- typedef **basic\_string**<\_CharT> **string\_type**

## Public Member Functions

- **moneypunct** (size\_t \_\_refs=0)
- **moneypunct** (\_\_cache\_type \* \_\_cache, size\_t \_\_refs=0)
- **moneypunct** (\_\_c\_locale \_\_cloc, const char \* \_\_s, size\_t \_\_refs=0)
- **string\_type curr\_symbol** () const
- **char\_type decimal\_point** () const
- int **frac\_digits** () const
- **string grouping** () const
- **string\_type negative\_sign** () const
- **string\_type positive\_sign** () const
- **char\_type thousands\_sep** () const
- pattern **pos\_format** () const
- pattern **neg\_format** () const

## Static Public Member Functions

- static pattern **\_S\_construct\_pattern** (char \_\_precedes, char \_\_space, char \_\_posn) throw ()

## Static Public Attributes

- static const char \* **\_S\_atoms**
- static const pattern **\_S\_default\_pattern**
- static **locale::id** **id**
- static const bool **intl**

## Protected Member Functions

- virtual **~moneypunct** ()
- void **\_M\_initialize\_moneypunct** (\_\_c\_locale \_\_cloc=0, const char \* \_\_name=0)
- template<> void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- template<> void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- template<> void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- template<> void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- virtual **string\_type do\_curr\_symbol** () const
- virtual **char\_type do\_decimal\_point** () const
- virtual int **do\_frac\_digits** () const
- virtual **string do\_grouping** () const
- virtual pattern **do\_neg\_format** () const



- virtual [string\\_type do\\_negative\\_sign](#) () const
- virtual pattern [do\\_pos\\_format](#) () const
- virtual [string\\_type do\\_positive\\_sign](#) () const
- virtual [char\\_type do\\_thousands\\_sep](#) () const

#### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

#### 4.641.1 Detailed Description

`template<typename _CharT, bool _Intl>class std::moneypunct< _CharT, _Intl >`

Primary class template moneypunct.

This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.

Definition at line 934 of file locale\_facets\_nonio.h.

#### 4.641.2 Member Typedef Documentation

4.641.2.1 `template<typename _CharT, bool _Intl> typedef _CharT std::moneypunct< _CharT, _Intl >::char_type`

Public typedefs.

Definition at line 940 of file locale\_facets\_nonio.h.

4.641.2.2 `template<typename _CharT, bool _Intl> typedef basic_string<_CharT> std::moneypunct< _CharT, _Intl >::string_type`

Public typedefs.

Definition at line 941 of file locale\_facets\_nonio.h.

#### 4.641.3 Constructor & Destructor Documentation

4.641.3.1 `template<typename _CharT, bool _Intl> std::moneypunct< _CharT, _Intl >::moneypunct ( size_t __refs = 0 )  
[inline], [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

---

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

Definition at line 963 of file locale\_facets\_nonio.h.

4.641.3.2 `template<typename _CharT, bool _Intl> std::moneypunct<_CharT, _Intl>::moneypunct ( __cache_type * __cache, size_t __refs = 0 ) [inline], [explicit]`

Constructor performs initialization.

This is an internal constructor.

Parameters

<code>__cache</code>	Cache for optimization.
<code>__refs</code>	Passed to the base facet class.

Definition at line 976 of file locale\_facets\_nonio.h.

4.641.3.3 `template<typename _CharT, bool _Intl> std::moneypunct<_CharT, _Intl>::moneypunct ( __c_locale __cloc, const char * __s, size_t __refs = 0 ) [inline], [explicit]`

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

Parameters

<code>__cloc</code>	The C locale.
<code>__s</code>	The name of a locale.
<code>__refs</code>	Passed to the base facet class.

Definition at line 991 of file locale\_facets\_nonio.h.

4.641.3.4 `template<typename _CharT, bool _Intl> virtual std::moneypunct<_CharT, _Intl>::~~moneypunct ( ) [protected], [virtual]`

Destructor.

#### 4.641.4 Member Function Documentation

4.641.4.1 `template<typename _CharT, bool _Intl> string_type std::moneypunct<_CharT, _Intl>::curr_symbol ( ) const [inline]`

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. It does so by returning `moneypunct<char, _Intl>::do_curr_symbol()`.

Returns

*string\_type* representing a currency symbol.

Definition at line 1061 of file locale\_facets\_nonio.h.

References `std::moneypunct<_CharT, _Intl>::do_curr_symbol()`.

4.641.4.2 `template<typename _CharT, bool _Intl> char_type std::moneypunct<_CharT, _Intl>::decimal_point ( ) const [inline]`

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning `money_punct<char_type>::do_decimal_point()`.

#### Returns

*char\_type* representing a decimal point.

Definition at line 1005 of file `locale_facets_nonio.h`.

References `std::money_punct<_CharT, _Intl>::do_decimal_point()`.

4.641.4.3 `template<typename _CharT, bool _Intl> virtual string_type std::money_punct<_CharT, _Intl>::do_curr_symbol ( ) const [inline], [protected], [virtual]`

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. This function is a hook for derived classes to change the value returned.

#### See also

`curr_symbol()` for details.

#### Returns

*string\_type* representing a currency symbol.

Definition at line 1207 of file `locale_facets_nonio.h`.

Referenced by `std::money_punct<_CharT, _Intl>::curr_symbol()`.

4.641.4.4 `template<typename _CharT, bool _Intl> virtual char_type std::money_punct<_CharT, _Intl>::do_decimal_point ( ) const [inline], [protected], [virtual]`

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

#### Returns

*char\_type* representing a decimal point.

Definition at line 1169 of file `locale_facets_nonio.h`.

Referenced by `std::money_punct<_CharT, _Intl>::decimal_point()`.

4.641.4.5 `template<typename _CharT, bool _Intl> virtual int std::money_punct<_CharT, _Intl>::do_frac_digits ( ) const [inline], [protected], [virtual]`

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

#### See also

`frac_digits()` for details.

**Returns**

Number of digits in amount fraction.

Definition at line 1247 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::frac\_digits().

4.641.4.6 `template<typename _CharT, bool _Intl> virtual string std::moneypunct< _CharT, _Intl >::do_grouping ( ) const`  
`[inline], [protected], [virtual]`

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

**See also**

grouping() for details.

**Returns**

String representing grouping specification.

Definition at line 1194 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::grouping().

4.641.4.7 `template<typename _CharT, bool _Intl> virtual pattern std::moneypunct< _CharT, _Intl >::do_neg_format ( ) const`  
`[inline], [protected], [virtual]`

Return pattern for money values.

This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

**See also**

neg\_format() for details.

**Returns**

Pattern for money values.

Definition at line 1275 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::neg\_format().

4.641.4.8 `template<typename _CharT, bool _Intl> virtual string_type std::moneypunct< _CharT, _Intl >::do_negative_sign ( ) const`  
`[inline], [protected], [virtual]`

Return negative sign string.

This function returns a string\_type to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

**See also**

negative\_sign() for details.

**Returns**

*string\_type* representing a negative sign.

Definition at line 1233 of file locale\_facets\_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl>::negative_sign()`.

**4.641.4.9** `template<typename _CharT, bool _Intl> virtual pattern std::moneypunct<_CharT, _Intl>::do_pos_format ( ) const [inline], [protected], [virtual]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

**See also**

`pos_format()` for details.

**Returns**

Pattern for money values.

Definition at line 1261 of file locale\_facets\_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl>::pos_format()`.

**4.641.4.10** `template<typename _CharT, bool _Intl> virtual string_type std::moneypunct<_CharT, _Intl>::do_positive_sign ( ) const [inline], [protected], [virtual]`

Return positive sign string.

This function returns a *string\_type* to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

**See also**

`positive_sign()` for details.

**Returns**

*string\_type* representing a positive sign.

Definition at line 1220 of file locale\_facets\_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl>::positive_sign()`.

**4.641.4.11** `template<typename _CharT, bool _Intl> virtual char_type std::moneypunct<_CharT, _Intl>::do_thousands_sep ( ) const [inline], [protected], [virtual]`

Return thousands separator character.

Returns a *char\_type* to use as a thousands separator. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a thousands separator.

Definition at line 1181 of file locale\_facets\_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl>::thousands_sep()`.

**4.641.4.12** `template<typename _CharT, bool _Intl> int std::moneypunct<_CharT, _Intl>::frac_digits ( ) const`  
`[inline]`

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning `std::moneypunct<char_type>::do_frac_digits()`.

The fractional part of a money amount is optional. But if it is present, there must be `frac_digits()` digits.

#### Returns

Number of digits in amount fraction.

Definition at line 1111 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_frac_digits()`.

**4.641.4.13** `template<typename _CharT, bool _Intl> string std::moneypunct<_CharT, _Intl>::grouping ( ) const`  
`[inline]`

Return grouping specification.

This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the `grouping()` returns `\003\002` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `32`, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `std::moneypunct<char_type>::do_grouping()`.

#### Returns

string representing grouping specification.

Definition at line 1048 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_grouping()`.

**4.641.4.14** `template<typename _CharT, bool _Intl> pattern std::moneypunct<_CharT, _Intl>::neg_format ( ) const`  
`[inline]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `std::moneypunct<char_type>::do_pos_format()` or `std::moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

#### Returns

Pattern for money values.

Definition at line 1151 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_neg_format()`.

**4.641.4.15** `template<typename _CharT, bool _Intl> string_type std::moneypunct<_CharT, _Intl>::negative_sign ( ) const`  
`[inline]`

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. It does so by returning `moneypunct<char_type>::do_negative_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `neg_format()` and the remainder appear at the end of the formatted string.

#### Returns

*string\_type* representing a negative sign.

Definition at line 1095 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_negative_sign()`.

**4.641.4.16** `template<typename _CharT, bool _Intl> pattern std::moneypunct<_CharT, _Intl>::pos_format ( ) const`  
`[inline]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

#### Returns

Pattern for money values.

Definition at line 1147 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_pos_format()`.

**4.641.4.17** `template<typename _CharT, bool _Intl> string_type std::moneypunct<_CharT, _Intl>::positive_sign ( ) const`  
`[inline]`

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. It does so by returning `moneypunct<char_type>::do_positive_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `pos_format()` and the remainder appear at the end of the formatted string.

#### Returns

*string\_type* representing a positive sign.

Definition at line 1078 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_positive_sign()`.

**4.641.4.18** `template<typename _CharT, bool _Intl> char_type std::moneypunct<_CharT, _Intl>::thousands_sep ( ) const`  
[inline]

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning `moneypunct<char_type>::do_thousands_sep()`.

#### Returns

`char_type` representing a thousands separator.

Definition at line 1018 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_thousands_sep()`.

### 4.641.5 Member Data Documentation

**4.641.5.1** `template<typename _CharT, bool _Intl> locale::id std::moneypunct<_CharT, _Intl>::id` [static]

Numpunct facet id.

Definition at line 953 of file `locale_facets_nonio.h`.

**4.641.5.2** `template<typename _CharT, bool _Intl> const bool std::moneypunct<_CharT, _Intl>::intl` [static]

This value is provided by the standard, but no reason for its existence.

Definition at line 951 of file `locale_facets_nonio.h`.

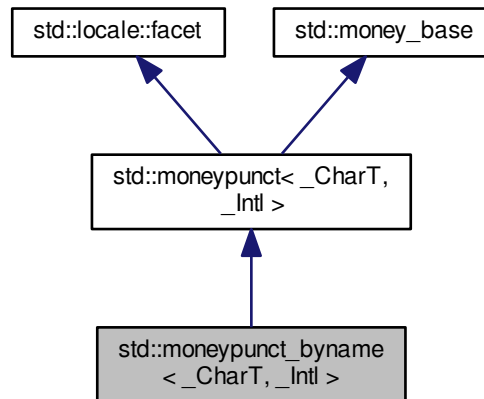
The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)



#### 4.642 `std::moneypunct_byname<_CharT, _Intl>` Class Template Reference

Inheritance diagram for `std::moneypunct_byname<_CharT, _Intl>`:



##### Public Types

- enum { **`_S_minus`**, **`_S_zero`**, **`_S_end`** }
- typedef `__moneypunct_cache<_CharT, _Intl>` **`__cache_type`**
- typedef `_CharT` **`char_type`**
- enum **`part`** { **`none`**, **`space`**, **`symbol`**, **`sign`**, **`value`** }
- typedef `basic_string<_CharT>` **`string_type`**

##### Public Member Functions

- **`moneypunct_byname`** (const char \*\_\_s, size\_t \_\_refs=0)
- `string_type curr_symbol` () const
- `char_type decimal_point` () const
- int `frac_digits` () const
- `string grouping` () const
- `string_type negative_sign` () const
- `string_type positive_sign` () const
- `char_type thousands_sep` () const
- pattern `pos_format` () const
- pattern `neg_format` () const

##### Static Public Member Functions

- static pattern **`_S_construct_pattern`** (char \_\_precedes, char \_\_space, char \_\_posn) throw ()

## Static Public Attributes

- static const char \* **\_S\_atoms**
- static const pattern **\_S\_default\_pattern**
- static [locale::id](#) **id**
- static const bool **intl**

## Protected Member Functions

- void **\_M\_initialize\_moneypunct** (\_\_c\_locale \_\_cloc=0, const char \*\_\_name=0)
- template<> void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- template<> void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- template<> void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- template<> void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- virtual [string\\_type](#) **do\_curr\_symbol** () const
- virtual [char\\_type](#) **do\_decimal\_point** () const
- virtual int **do\_frac\_digits** () const
- virtual [string](#) **do\_grouping** () const
- virtual pattern **do\_neg\_format** () const
- virtual [string\\_type](#) **do\_negative\_sign** () const
- virtual pattern **do\_pos\_format** () const
- virtual [string\\_type](#) **do\_positive\_sign** () const
- virtual [char\\_type](#) **do\_thousands\_sep** () const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## 4.642.1 Detailed Description

template<typename \_CharT, bool \_Intl> class std::moneypunct\_byname<\_CharT, \_Intl>

class moneypunct\_byname [22.2.6.4].

Definition at line 1324 of file locale\_facets\_nonio.h.

## 4.642.2 Member Function Documentation

4.642.2.1 template<typename \_CharT, bool \_Intl> [string\\_type](#) std::moneypunct<\_CharT, \_Intl>::curr\_symbol ( ) const  
[inline], [inherited]

Return currency symbol string.

This function returns a [string\\_type](#) to use as a currency symbol. It does so by returning returning moneypunct<char↵\_type>::do\_curr\_symbol().

**Returns**

*string\_type* representing a currency symbol.

Definition at line 1061 of file locale\_facets\_nonio.h.

References `std::moneypunct<_CharT, _Intl>::do_curr_symbol()`.

**4.642.2.2** `template<typename _CharT, bool _Intl> char_type std::moneypunct<_CharT, _Intl>::decimal_point ( ) const`  
`[inline], [inherited]`

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning `moneypunct<char_type>::do_decimal_point()`.

**Returns**

*char\_type* representing a decimal point.

Definition at line 1005 of file locale\_facets\_nonio.h.

References `std::moneypunct<_CharT, _Intl>::do_decimal_point()`.

**4.642.2.3** `template<typename _CharT, bool _Intl> virtual string_type std::moneypunct<_CharT, _Intl>::do_curr_symbol ( ) const`  
`[inline], [protected], [virtual], [inherited]`

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. This function is a hook for derived classes to change the value returned.

**See also**

`curr_symbol()` for details.

**Returns**

*string\_type* representing a currency symbol.

Definition at line 1207 of file locale\_facets\_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl>::curr_symbol()`.

**4.642.2.4** `template<typename _CharT, bool _Intl> virtual char_type std::moneypunct<_CharT, _Intl>::do_decimal_point ( ) const`  
`[inline], [protected], [virtual], [inherited]`

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a decimal point.

Definition at line 1169 of file locale\_facets\_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl>::decimal_point()`.

**4.642.2.5** `template<typename _CharT, bool _Intl> virtual int std::moneypunct< _CharT, _Intl >::do_frac_digits ( ) const`  
`[inline], [protected], [virtual], [inherited]`

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

See also

frac\_digits() for details.

Returns

Number of digits in amount fraction.

Definition at line 1247 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::frac\_digits().

**4.642.2.6** `template<typename _CharT, bool _Intl> virtual string std::moneypunct< _CharT, _Intl >::do_grouping ( ) const`  
`[inline], [protected], [virtual], [inherited]`

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

See also

grouping() for details.

Returns

String representing grouping specification.

Definition at line 1194 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::grouping().

**4.642.2.7** `template<typename _CharT, bool _Intl> virtual pattern std::moneypunct< _CharT, _Intl >::do_neg_format ( ) const`  
`[inline], [protected], [virtual], [inherited]`

Return pattern for money values.

This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

See also

neg\_format() for details.

Returns

Pattern for money values.

Definition at line 1275 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::neg\_format().

4.642.2.8 `template<typename _CharT, bool _Intl> virtual string_type std::moneypunct<_CharT, _Intl>::do_negative_sign ( ) const` `[inline], [protected], [virtual], [inherited]`

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

**See also**

`negative_sign()` for details.

**Returns**

*string\_type* representing a negative sign.

Definition at line 1233 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl>::negative_sign()`.

4.642.2.9 `template<typename _CharT, bool _Intl> virtual pattern std::moneypunct<_CharT, _Intl>::do_pos_format ( ) const` `[inline], [protected], [virtual], [inherited]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

**See also**

`pos_format()` for details.

**Returns**

Pattern for money values.

Definition at line 1261 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl>::pos_format()`.

4.642.2.10 `template<typename _CharT, bool _Intl> virtual string_type std::moneypunct<_CharT, _Intl>::do_positive_sign ( ) const` `[inline], [protected], [virtual], [inherited]`

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

**See also**

`positive_sign()` for details.

**Returns**

*string\_type* representing a positive sign.

Definition at line 1220 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl>::positive_sign()`.

4.642.2.11 `template<typename _CharT, bool _Intl> virtual char_type std::moneypunct< _CharT, _Intl >::do_thousands_sep ( ) const` `[inline], [protected], [virtual], [inherited]`

Return thousands separator character.

Returns a `char_type` to use as a thousands separator. This function is a hook for derived classes to change the value returned.

#### Returns

*char\_type* representing a thousands separator.

Definition at line 1181 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::thousands_sep()`.

4.642.2.12 `template<typename _CharT, bool _Intl> int std::moneypunct< _CharT, _Intl >::frac_digits ( ) const` `[inline], [inherited]`

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning `moneypunct<char_type>::do_frac_digits()`.

The fractional part of a money amount is optional. But if it is present, there must be `frac_digits()` digits.

#### Returns

Number of digits in amount fraction.

Definition at line 1111 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_frac_digits()`.

4.642.2.13 `template<typename _CharT, bool _Intl> string std::moneypunct< _CharT, _Intl >::grouping ( ) const` `[inline], [inherited]`

Return grouping specification.

This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the `grouping()` returns `\003\002` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `32`, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `moneypunct<char_type>::do_grouping()`.

#### Returns

string representing grouping specification.

Definition at line 1048 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_grouping()`.

**4.642.2.14** `template<typename _CharT, bool _Intl> pattern std::moneypunct<_CharT, _Intl>::neg_format ( ) const`  
`[inline], [inherited]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is \$+10.01.

#### Returns

Pattern for money values.

Definition at line 1151 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_neg_format()`.

**4.642.2.15** `template<typename _CharT, bool _Intl> string_type std::moneypunct<_CharT, _Intl>::negative_sign ( ) const`  
`[inline], [inherited]`

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. It does so by returning `moneypunct<char_type>::do_negative_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `neg_format()` and the remainder appear at the end of the formatted string.

#### Returns

*string\_type* representing a negative sign.

Definition at line 1095 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_negative_sign()`.

**4.642.2.16** `template<typename _CharT, bool _Intl> pattern std::moneypunct<_CharT, _Intl>::pos_format ( ) const`  
`[inline], [inherited]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates

0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and pos\_format() pattern {symbol,sign,value,none}, curr\_symbol() == '\$' positive\_sign() == '+', and value 10.01, and options set to force the symbol, the corresponding string is \$+10.01.

#### Returns

Pattern for money values.

Definition at line 1147 of file locale\_facets\_nonio.h.

References std::moneypunct< \_CharT, \_Intl >::do\_pos\_format().

**4.642.2.17** `template<typename _CharT, bool _Intl> string_type std::moneypunct< _CharT, _Intl >::positive_sign ( ) const`  
`[inline], [inherited]`

Return positive sign string.

This function returns a string\_type to use as a sign for positive amounts. It does so by returning returning moneypunct<char\_type>::do\_positive\_sign().

If the return value contains more than one character, the first character appears in the position indicated by pos\_format() and the remainder appear at the end of the formatted string.

#### Returns

*string\_type* representing a positive sign.

Definition at line 1078 of file locale\_facets\_nonio.h.

References std::moneypunct< \_CharT, \_Intl >::do\_positive\_sign().

**4.642.2.18** `template<typename _CharT, bool _Intl> char_type std::moneypunct< _CharT, _Intl >::thousands_sep ( ) const`  
`[inline], [inherited]`

Return thousands separator character.

This function returns a char\_type to use as a thousands separator. It does so by returning returning moneypunct<char\_type>::do\_thousands\_sep().

#### Returns

char\_type representing a thousands separator.

Definition at line 1018 of file locale\_facets\_nonio.h.

References std::moneypunct< \_CharT, \_Intl >::do\_thousands\_sep().

### 4.642.3 Member Data Documentation

**4.642.3.1** `template<typename _CharT, bool _Intl> locale::id std::moneypunct< _CharT, _Intl >::id` `[static],`  
`[inherited]`

Numpunct facet id.

Definition at line 953 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)



#### 4.643 `std::move_iterator<_Iterator>` Class Template Reference

##### Public Types

- typedef `__traits_type::difference_type` **difference\_type**
- typedef `__traits_type::iterator_category` **iterator\_category**
- typedef `_Iterator` **iterator\_type**
- typedef `_Iterator` **pointer**
- typedef `value_type &&` **reference**
- typedef `__traits_type::value_type` **value\_type**

##### Public Member Functions

- **move\_iterator** (`iterator_type __i`)
- template<typename `_Iter`> **move\_iterator** (const [move\\_iterator](#)< `_Iter` > &`__i`)
- `iterator_type` **base** () const
- reference **operator\*** () const
- [move\\_iterator](#) **operator+** (`difference_type __n`) const
- [move\\_iterator](#) & **operator++** ()
- [move\\_iterator](#) **operator++** (`int`)
- [move\\_iterator](#) & **operator+=** (`difference_type __n`)
- [move\\_iterator](#) **operator-** (`difference_type __n`) const
- [move\\_iterator](#) & **operator--** ()
- [move\\_iterator](#) **operator--** (`int`)
- [move\\_iterator](#) & **operator-=** (`difference_type __n`)
- pointer **operator->** () const
- reference **operator[]** (`difference_type __n`) const

##### Protected Types

- typedef `iterator_traits<_Iterator>` **\_\_traits\_type**

##### Protected Attributes

- `_Iterator` **\_M\_current**

##### 4.643.1 Detailed Description

template<typename `_Iterator`>class `std::move_iterator<_Iterator>`

Class template `move_iterator` is an iterator adapter with the same behavior as the underlying iterator except that its dereference operator implicitly converts the value returned by the underlying iterator's dereference operator to an rvalue reference. Some generic algorithms can be called with move iterators to replace copying with moving.

Definition at line 411 of file `cpp_type_traits.h`.

The documentation for this class was generated from the following files:

- [cpp\\_type\\_traits.h](#)
- [stl\\_iterator.h](#)

## 4.644 std::multimap&lt; \_Key, \_Tp, \_Compare, \_Alloc &gt; Class Template Reference

## Public Types

- typedef \_Alloc **allocator\_type**
- typedef \_Rep\_type::const\_iterator **const\_iterator**
- typedef \_Pair\_alloc\_type::const\_pointer **const\_pointer**
- typedef \_Pair\_alloc\_type::const\_reference **const\_reference**
- typedef \_Rep\_type::const\_reverse\_iterator **const\_reverse\_iterator**
- typedef \_Rep\_type::difference\_type **difference\_type**
- typedef \_Rep\_type::iterator **iterator**
- typedef \_Compare **key\_compare**
- typedef \_Key **key\_type**
- typedef \_Tp **mapped\_type**
- typedef \_Pair\_alloc\_type::pointer **pointer**
- typedef \_Pair\_alloc\_type::reference **reference**
- typedef \_Rep\_type::reverse\_iterator **reverse\_iterator**
- typedef \_Rep\_type::size\_type **size\_type**
- typedef std::pair< const \_Key, \_Tp > **value\_type**

## Public Member Functions

- **multimap** ()
- **multimap** (const \_Compare &\_\_comp, const allocator\_type &\_\_a=allocator\_type())
- **multimap** (const **multimap** &\_\_x)
- **multimap** (**multimap** &&\_\_x) noexcept(is\_nothrow\_copy\_constructible< \_Compare >::value)
- **multimap** (initializer\_list< **value\_type** > \_\_l, const \_Compare &\_\_comp=\_Compare(), const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator > **multimap** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_InputIterator > **multimap** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_comp, const allocator\_type &\_\_a=allocator\_type())
- iterator **begin** () noexcept
- const\_iterator **begin** () const noexcept
- const\_iterator **cbegin** () const noexcept
- const\_iterator **cend** () const noexcept
- void **clear** () noexcept
- size\_type **count** (const key\_type &\_\_x) const
- const\_reverse\_iterator **crbegin** () const noexcept
- const\_reverse\_iterator **crend** () const noexcept
- template<typename... \_Args> iterator **emplace** (\_Args &&...\_\_args)
- template<typename... \_Args> iterator **emplace\_hint** (const\_iterator \_\_pos, \_Args &&...\_\_args)
- bool **empty** () const noexcept
- iterator **end** () noexcept
- const\_iterator **end** () const noexcept
- std::pair< iterator, iterator > **equal\_range** (const key\_type &\_\_x)
- std::pair< const\_iterator, const\_iterator > **equal\_range** (const key\_type &\_\_x) const
- iterator **erase** (const\_iterator \_\_position)
- \_GLIBCXX\_ABI\_TAG\_CXX11 iterator **erase** (iterator \_\_position)
- size\_type **erase** (const key\_type &\_\_x)
- iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- iterator **find** (const key\_type &\_\_x)

- const\_iterator [find](#) (const key\_type &\_\_x) const
- allocator\_type [get\\_allocator](#) () const noexcept
- iterator [insert](#) (const value\_type &\_\_x)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type> iterator **insert** (\_Pair &&\_\_x)
- iterator [insert](#) (const\_iterator \_\_position, const value\_type &\_\_x)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type> iterator **insert** (const\_iterator \_\_position, \_Pair &&\_\_x)
- template<typename \_InputIterator > void [insert](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void [insert](#) (initializer\_list< value\_type > \_\_l)
- key\_compare [key\\_comp](#) () const
- iterator [lower\\_bound](#) (const key\_type &\_\_x)
- const\_iterator [lower\\_bound](#) (const key\_type &\_\_x) const
- size\_type [max\\_size](#) () const noexcept
- [multimap](#) & [operator=](#) (const [multimap](#) &\_\_x)
- [multimap](#) & [operator=](#) ([multimap](#) &&\_\_x)
- [multimap](#) & [operator=](#) (initializer\_list< value\_type > \_\_l)
- [reverse\\_iterator](#) [rbegin](#) () noexcept
- [const\\_reverse\\_iterator](#) [rbegin](#) () const noexcept
- [reverse\\_iterator](#) [rend](#) () noexcept
- [const\\_reverse\\_iterator](#) [rend](#) () const noexcept
- size\_type [size](#) () const noexcept
- void [swap](#) ([multimap](#) &\_\_x)
- iterator [upper\\_bound](#) (const key\_type &\_\_x)
- const\_iterator [upper\\_bound](#) (const key\_type &\_\_x) const
- value\_compare [value\\_comp](#) () const

#### Friends

- template<typename \_K1, typename \_T1, typename \_C1, typename \_A1 > bool **operator**< (const [multimap](#)< \_K1, \_T1, \_C1, \_A1 > &, const [multimap](#)< \_K1, \_T1, \_C1, \_A1 > &)
- template<typename \_K1, typename \_T1, typename \_C1, typename \_A1 > bool **operator==** (const [multimap](#)< \_K1, \_T1, \_C1, \_A1 > &, const [multimap](#)< \_K1, \_T1, \_C1, \_A1 > &)

#### 4.644.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> class std::multimap<_Key, _Tp, _Compare, _Alloc>
```

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

#### Template Parameters

<a href="#">_Key</a>	Type of key objects.
<a href="#">_Tp</a>	Type of mapped objects.
<a href="#">_Compare</a>	Comparison function object type, defaults to less<_Key>.
<a href="#">_Alloc</a>	Allocator type, defaults to allocator<pair<const _Key, _Tp>>.

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using equivalent keys). For a `multimap<Key, T>` the `key_type` is `Key`, the `mapped_type` is `T`, and the `value_type` is `std::pair<const Key, T>`.

Multimaps support bidirectional iterators.

The private tree data is declared exactly the same way for map and multimap; the distinction is made entirely in how the tree functions are called (\*\_unique versus \*\_equal, same as the standard).

Definition at line 95 of file stl\_multimap.h.

#### 4.644.2 Constructor & Destructor Documentation

4.644.2.1 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap ( )`  
[inline]

Default constructor creates no elements.

Definition at line 157 of file stl\_multimap.h.

4.644.2.2 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap ( const _Compare & __comp, const allocator_type & __a = allocator_type() )` [inline], [explicit]

Creates a multimap with no elements.

##### Parameters

<code>__comp</code>	A comparison object.
<code>__a</code>	An allocator object.

Definition at line 166 of file stl\_multimap.h.

4.644.2.3 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap ( const multimap<_Key, _Tp, _Compare, _Alloc> & __x )` [inline]

Multimap copy constructor.

##### Parameters

<code>__x</code>	A multimap of identical element and allocator types.
------------------	--

The newly-created multimap uses a copy of the allocation object used by `__x`.

Definition at line 177 of file stl\_multimap.h.

4.644.2.4 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap ( multimap<_Key, _Tp, _Compare, _Alloc> && __x )` [inline], [noexcept]

Multimap move constructor.

##### Parameters

<code>__x</code>	A multimap of identical element and allocator types.
------------------	--

The newly-created multimap contains the exact contents of `__x`. The contents of `__x` are a valid, but unspecified multimap.

Definition at line 188 of file stl\_multimap.h.

```
4.644.2.5  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =  
            std::allocator<std::pair<const _Key, _Tp> >> std::multimap<_Key, _Tp, _Compare, _Alloc >::multimap  
            ( initializer_list< value_type > __l, const _Compare & __comp = _Compare(), const allocator_type & __a =  
              allocator_type() ) [inline]
```

Builds a multimap from an initializer\_list.

## Parameters

<code>__l</code>	An initializer_list.
<code>__comp</code>	A comparison functor.
<code>__a</code>	An allocator object.

Create a multimap consisting of copies of the elements from the initializer\_list. This is linear in N if the list is already sorted, and NlogN otherwise (where N is `__l.size()`).

Definition at line 202 of file `stl_multimap.h`.

```
4.644.2.6 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> template<typename _InputIterator > std::multimap< _Key, _Tp,
_Compare, _Alloc >::multimap ( _InputIterator __first, _InputIterator __last ) [inline]
```

Builds a multimap from a range.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

Create a multimap consisting of copies of the elements from [`__first`,`__last`). This is linear in N if the range is already sorted, and NlogN otherwise (where N is `distance(__first, __last)`).

Definition at line 219 of file `stl_multimap.h`.

```
4.644.2.7 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> template<typename _InputIterator > std::multimap< _Key, _Tp,
_Compare, _Alloc >::multimap ( _InputIterator __first, _InputIterator __last, const _Compare & __comp, const
allocator_type & __a = allocator_type() ) [inline]
```

Builds a multimap from a range.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__comp</code>	A comparison functor.
<code>__a</code>	An allocator object.

Create a multimap consisting of copies of the elements from [`__first`,`__last`). This is linear in N if the range is already sorted, and NlogN otherwise (where N is `distance(__first, __last)`).

Definition at line 235 of file `stl_multimap.h`.

## 4.644.3 Member Function Documentation

```
4.644.3.1 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::begin ( )
[inline], [noexcept]
```

Returns a read/write iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 314 of file `stl_multimap.h`.

```
4.644.3.2 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::begin
( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 323 of file stl\_multimap.h.

```
4.644.3.3 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::cbegin
( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 387 of file stl\_multimap.h.

```
4.644.3.4 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::cend
( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 396 of file stl\_multimap.h.

```
4.644.3.5 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> void std::multimap<_Key, _Tp, _Compare, _Alloc>::clear ( )
[inline], [noexcept]
```

Erases all elements in a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 697 of file stl\_multimap.h.

Referenced by `std::multimap<_Key, _Tp, _Compare, _Alloc>::operator=()`.

```
4.644.3.6 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> size_type std::multimap<_Key, _Tp, _Compare, _Alloc>::count (
const key_type & __x ) const [inline]
```

Finds the number of elements with given key.

#### Parameters

<code>__x</code>	Key of (key, value) pairs to be located.
------------------	--

#### Returns

Number of elements with specified key.

Definition at line 754 of file stl\_multimap.h.

```
4.644.3.7 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::multimap<_Key, _Tp, _Compare,
_Alloc>::crbegin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 405 of file stl\_multimap.h.

```
4.644.3.8 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> const_reverse_iterator std::multimap<_Key, _Tp, _Compare,
_Alloc>::crend( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 414 of file stl\_multimap.h.

```
4.644.3.9 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> template<typename... _Args> iterator std::multimap<_Key, _Tp,
_Compare, _Alloc>::emplace( _Args &&... _args ) [inline]
```

Build and insert a std::pair into the multimap.

Parameters

<code>__args</code>	Arguments used to generate a new pair instance (see std::piecewise_construct for passing arguments to each part of the pair constructor).
---------------------	---

Returns

An iterator that points to the inserted (key,value) pair.

This function builds and inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.

Insertion requires logarithmic time.

Definition at line 454 of file stl\_multimap.h.

```
4.644.3.10 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> template<typename... _Args> iterator std::multimap<_Key, _Tp,
_Compare, _Alloc>::emplace_hint( const_iterator __pos, _Args &&... _args ) [inline]
```

Builds and inserts a std::pair into the multimap.

Parameters

<code>__pos</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__args</code>	Arguments used to generate a new pair instance (see std::piecewise_construct for passing arguments to each part of the pair constructor).

Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html>.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 481 of file stl\_multimap.h.



```
4.644.3.11 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> bool std::multimap<_Key, _Tp, _Compare, _Alloc>::empty ( )
const [inline], [noexcept]
```

Returns true if the multimap is empty.

Definition at line 421 of file stl\_multimap.h.

```
4.644.3.12 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::end ( )
[inline], [noexcept]
```

Returns a read/write iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 332 of file stl\_multimap.h.

```
4.644.3.13 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::end (
) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 341 of file stl\_multimap.h.

```
4.644.3.14 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::pair<iterator, iterator> std::multimap<_Key, _Tp, _Compare,
_Alloc>::equal_range ( const key_type & __x ) [inline]
```

Finds a subsequence matching given key.

Parameters

<code>__x</code>	Key of (key, value) pairs to be located.
------------------	--

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
c.upper_bound(val))
```

(but is faster than making the calls separately).

Definition at line 821 of file stl\_multimap.h.

```
4.644.3.15 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::pair<const_iterator, const_iterator> std::multimap<_Key,
_Tp, _Compare, _Alloc>::equal_range ( const key_type & __x ) const [inline]
```

Finds a subsequence matching given key.

## Parameters

<code>__x</code>	Key of (key, value) pairs to be located.
------------------	--

## Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

Definition at line 838 of file stl\_multimap.h.

```
4.644.3.16 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::erase (
const_iterator __position ) [inline]
```

Erases an element from a multimap.

## Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

## Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 595 of file stl\_multimap.h.

```
4.644.3.17 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::erase (
const key_type & __x ) [inline]
```

Erases elements according to the provided key.

## Parameters

<code>__x</code>	Key of element to be erased.
------------------	------------------------------

## Returns

The number of elements erased.

This function erases all elements located by the given key from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 631 of file stl\_multimap.h.

4.644.3.18 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =  
std::allocator<std::pair<const _Key, _Tp> >> iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::erase (  
const_iterator __first, const_iterator __last ) [inline]`

Erases a [first,last) range of elements from a multimap.

## Parameters

<code>__first</code>	Iterator pointing to the start of the range to be erased.
<code>__last</code>	Iterator pointing to the end of the range to be erased .

## Returns

The iterator `__last`.

This function erases a sequence of elements from a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 652 of file `stl_multimap.h`.

```
4.644.3.19  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::find ( const
            key_type & __x ) [inline]
```

Tries to locate an element in a multimap.

## Parameters

<code>__x</code>	Key of (key, value) pair to be located.
------------------	---

## Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 730 of file `stl_multimap.h`.

```
4.644.3.20  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::find (
            const key_type & __x ) const [inline]
```

Tries to locate an element in a multimap.

## Parameters

<code>__x</code>	Key of (key, value) pair to be located.
------------------	---

## Returns

Read-only (constant) iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 745 of file `stl_multimap.h`.

```
4.644.3.21  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> allocator_type std::multimap<_Key, _Tp, _Compare, _Alloc>
            >::get_allocator ( ) const [inline], [noexcept]
```

Get a copy of the memory allocation object.

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

4.644.3.22 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =  
std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::insert (  
const value_type & __x ) [inline]`

Inserts a `std::pair` into the `multimap`.

## Parameters

<code>__x</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).
------------------	---

## Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a `std::map` the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.

Insertion requires logarithmic time.

Definition at line 501 of file `stl_multimap.h`.

Referenced by `std::multimap< _Key, _Tp, _Compare, _Alloc >::insert()`, and `std::multimap< _Key, _Tp, _Compare, _Alloc >::operator=()`.

```
4.644.3.23 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
const_iterator __position, const value_type &__x ) [inline]
```

Inserts a `std::pair` into the multimap.

## Parameters

<code>__position</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__x</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).

## Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a `std::map` the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html>.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 535 of file `stl_multimap.h`.

```
4.644.3.24 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> template<typename _InputIterator > void std::multimap< _Key,
_Tp, _Compare, _Alloc >::insert ( _InputIterator __first, _InputIterator __last ) [inline]
```

A template function that attempts to insert a range of elements.

## Parameters

<code>__first</code>	Iterator pointing to the start of the range to be inserted.
<code>__last</code>	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 562 of file `stl_multimap.h`.

```
4.644.3.25  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =  
            std::allocator<std::pair<const _Key, _Tp> >> void std::multimap<_Key, _Tp, _Compare, _Alloc >::insert (  
            initializer_list< value_type > __l ) [inline]
```

Attempts to insert a list of std::pairs into the multimap.

## Parameters

__l	A std::initializer_list<value_type> of pairs to be inserted.
-----	--

Complexity similar to that of the range constructor.

Definition at line 574 of file stl\_multimap.h.

References std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >::insert().

```
4.644.3.26 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> key_compare std::multimap< _Key, _Tp, _Compare, _Alloc
>::key_comp ( ) const [inline]
```

Returns the key comparison object out of which the multimap was constructed.

Definition at line 706 of file stl\_multimap.h.

```
4.644.3.27 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap< _Key, _Tp, _Compare, _Alloc
>::lower_bound ( const key_type & __x ) [inline]
```

Finds the beginning of a subsequence matching given key.

## Parameters

__x	Key of (key, value) pair to be located.
-----	---

## Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 769 of file stl\_multimap.h.

```
4.644.3.28 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc
>::lower_bound ( const key_type & __x ) const [inline]
```

Finds the beginning of a subsequence matching given key.

## Parameters

__x	Key of (key, value) pair to be located.
-----	---

## Returns

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful the iterator will point to the next greatest element or, if no such greater element exists, to end().

Definition at line 784 of file stl\_multimap.h.

```
4.644.3.29 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::max_size
( ) const [inline], [noexcept]
```

Returns the maximum size of the multimap.



Definition at line 431 of file `stl_multimap.h`.

```
4.644.3.30 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> multimap& std::multimap<_Key, _Tp, _Compare, _Alloc
>::operator= ( const multimap<_Key, _Tp, _Compare, _Alloc> & __x ) [inline]
```

Multimap assignment operator.

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Parameters

<code>__x</code>	A multimap of identical element and allocator types.
------------------	--

All the elements of `__x` are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 258 of file `stl_multimap.h`.

```
4.644.3.31 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> multimap& std::multimap<_Key, _Tp, _Compare, _Alloc
>::operator= ( multimap<_Key, _Tp, _Compare, _Alloc> && __x ) [inline]
```

Multimap move assignment operator.

Parameters

<code>__x</code>	A multimap of identical element and allocator types.
------------------	--

The contents of `__x` are moved into this multimap (without copying). `__x` is a valid, but unspecified multimap.

Definition at line 273 of file `stl_multimap.h`.

References `std::multimap<_Key, _Tp, _Compare, _Alloc>::clear()`, and `std::multimap<_Key, _Tp, _Compare, _Alloc>::swap()`.

```
4.644.3.32 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> multimap& std::multimap<_Key, _Tp, _Compare, _Alloc
>::operator= ( initializer_list<value_type> __l ) [inline]
```

Multimap list assignment operator.

Parameters

<code>__l</code>	An <code>initializer_list</code> .
------------------	------------------------------------

This function fills a multimap with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the multimap and that the resulting multimap's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 294 of file `stl_multimap.h`.

References `std::multimap<_Key, _Tp, _Compare, _Alloc>::clear()`, and `std::multimap<_Key, _Tp, _Compare, _Alloc>::insert()`.

```
4.644.3.33 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> reverse_iterator std::multimap<_Key, _Tp, _Compare, _Alloc
>::rbegin ( ) [inline],[noexcept]
```

Returns a read/write reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 350 of file `stl_multimap.h`.

```
4.644.3.34 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::multimap< _Key, _Tp, _Compare,
_Alloc >::rbegin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 359 of file stl\_multimap.h.

```
4.644.3.35 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc
>::rend ( ) [inline], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 368 of file stl\_multimap.h.

```
4.644.3.36 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::multimap< _Key, _Tp, _Compare,
_Alloc >::rend ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 377 of file stl\_multimap.h.

```
4.644.3.37 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::size ( )
const [inline], [noexcept]
```

Returns the size of the multimap.

Definition at line 426 of file stl\_multimap.h.

```
4.644.3.38 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> void std::multimap< _Key, _Tp, _Compare, _Alloc >::swap (
multimap< _Key, _Tp, _Compare, _Alloc > &__x ) [inline]
```

Swaps data with another multimap.

Parameters

__x	A multimap of the same element and allocator types.
-----	---

This exchanges the elements between two multimaps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

Definition at line 687 of file stl\_multimap.h.

Referenced by `std::multimap< _Key, _Tp, _Compare, _Alloc >::operator=()`, and `std::swap()`.

```
4.644.3.39 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap< _Key, _Tp, _Compare, _Alloc
>::upper_bound ( const key_type &__x ) [inline]
```

Finds the end of a subsequence matching given key.

## Parameters

<code>__x</code>	Key of (key, value) pair to be located.
------------------	---

## Returns

Iterator pointing to the first element greater than key, or end().

Definition at line 794 of file `stl_multimap.h`.

```
4.644.3.40 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc
>::upper_bound ( const key_type & __x ) const    [inline]
```

Finds the end of a subsequence matching given key.

## Parameters

<code>__x</code>	Key of (key, value) pair to be located.
------------------	---

## Returns

Read-only (constant) iterator pointing to first iterator greater than key, or end().

Definition at line 804 of file `stl_multimap.h`.

```
4.644.3.41 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> value_compare std::multimap<_Key, _Tp, _Compare, _Alloc
>::value_comp ( ) const    [inline]
```

Returns a value comparison object, built from the key comparison object out of which the multimap was constructed.

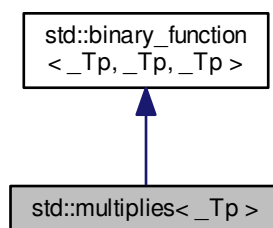
Definition at line 714 of file `stl_multimap.h`.

The documentation for this class was generated from the following file:

- [stl\\_multimap.h](#)

## 4.645 `std::multiplies<_Tp>` Struct Template Reference

Inheritance diagram for `std::multiplies<_Tp>`:



## Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Tp` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

## Public Member Functions

- `_Tp operator()` (`const _Tp &__x`, `const _Tp &__y`) `const`

## 4.645.1 Detailed Description

`template<typename _Tp>struct std::multiplies<_Tp>`

One of the [math functors](#).

Definition at line 158 of file `stl_function.h`.

## 4.645.2 Member Typedef Documentation

4.645.2.1 `typedef _Tp std::binary_function<_Tp,_Tp,_Tp>::first_argument_type` `[inherited]`

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.645.2.2 `typedef _Tp std::binary_function<_Tp,_Tp,_Tp>::result_type` `[inherited]`

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.645.2.3 `typedef _Tp std::binary_function<_Tp,_Tp,_Tp>::second_argument_type` `[inherited]`

`second_argument_type` is the type of the second argument

Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

4.646 `std::multiset<_Key,_Compare,_Alloc>` Class Template Reference

## Public Types

- typedef `_Alloc` `allocator_type`
- typedef `_Rep_type::const_iterator` `const_iterator`
- typedef `_Key_alloc_type::const_pointer` `const_pointer`
- typedef `_Key_alloc_type::const_reference` `const_reference`
- typedef `_Rep_type::const_reverse_iterator` `const_reverse_iterator`
- typedef `_Rep_type::difference_type` `difference_type`
- typedef `_Rep_type::const_iterator` `iterator`
- typedef `_Compare` `key_compare`

- typedef `_Key` **key\_type**
- typedef `_Key_alloc_type::pointer` **pointer**
- typedef `_Key_alloc_type::reference` **reference**
- typedef `_Rep_type::const_reverse_iterator` **reverse\_iterator**
- typedef `_Rep_type::size_type` **size\_type**
- typedef `_Compare` **value\_compare**
- typedef `_Key` **value\_type**

#### Public Member Functions

- **multiset** ()
- **multiset** (const `_Compare` &\_\_comp, const `allocator_type` &\_\_a=allocator\_type())
- template<typename `_InputIterator` > **multiset** (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- template<typename `_InputIterator` > **multiset** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp, const `allocator_type` &\_\_a=allocator\_type())
- **multiset** (const **multiset** &\_\_x)
- **multiset** (**multiset** &&\_\_x) noexcept(is\_nothrow\_copy\_constructible< `_Compare` >::value)
- **multiset** (initializer\_list< `value_type` > \_\_l, const `_Compare` &\_\_comp=\_Compare(), const `allocator_type` &\_\_a=allocator\_type())
- iterator **begin** () const noexcept
- iterator **cbegin** () const noexcept
- iterator **cend** () const noexcept
- void **clear** () noexcept
- `size_type` **count** (const `key_type` &\_\_x) const
- `reverse_iterator` **crbegin** () const noexcept
- `reverse_iterator` **crend** () const noexcept
- template<typename... `_Args`> iterator **emplace** (`_Args` &&...\_\_args)
- template<typename... `_Args`> iterator **emplace\_hint** (const\_iterator \_\_pos, `_Args` &&...\_\_args)
- bool **empty** () const noexcept
- iterator **end** () const noexcept
- `_GLIBCXX_ABI_TAG_CXX11` iterator **erase** (const\_iterator \_\_position)
- `size_type` **erase** (const `key_type` &\_\_x)
- `_GLIBCXX_ABI_TAG_CXX11` iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- `allocator_type` **get\_allocator** () const noexcept
- iterator **insert** (const `value_type` &\_\_x)
- iterator **insert** (`value_type` &&\_\_x)
- iterator **insert** (const\_iterator \_\_position, const `value_type` &\_\_x)
- iterator **insert** (const\_iterator \_\_position, `value_type` &&\_\_x)
- template<typename `_InputIterator` > void **insert** (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- void **insert** (initializer\_list< `value_type` > \_\_l)
- `key_compare` **key\_comp** () const
- `size_type` **max\_size** () const noexcept
- **multiset** & **operator=** (const **multiset** &\_\_x)
- **multiset** & **operator=** (**multiset** &&\_\_x)
- **multiset** & **operator=** (initializer\_list< `value_type` > \_\_l)
- `reverse_iterator` **rbegin** () const noexcept
- `reverse_iterator` **rend** () const noexcept
- `size_type` **size** () const noexcept
- void **swap** (**multiset** &\_\_x)
- `value_compare` **value\_comp** () const

- iterator [find](#) (const key\_type &\_\_x)
- const\_iterator [find](#) (const key\_type &\_\_x) const
- iterator [lower\\_bound](#) (const key\_type &\_\_x)
- const\_iterator [lower\\_bound](#) (const key\_type &\_\_x) const
- iterator [upper\\_bound](#) (const key\_type &\_\_x)
- const\_iterator [upper\\_bound](#) (const key\_type &\_\_x) const
- [std::pair](#)< iterator, iterator > [equal\\_range](#) (const key\_type &\_\_x)
- [std::pair](#)< const\_iterator, const\_iterator > [equal\\_range](#) (const key\_type &\_\_x) const

#### Friends

- template<typename \_K1, typename \_C1, typename \_A1 > bool **operator**< (const [multiset](#)< \_K1, \_C1, \_A1 > &, const [multiset](#)< \_K1, \_C1, \_A1 > &)
- template<typename \_K1, typename \_C1, typename \_A1 > bool **operator**== (const [multiset](#)< \_K1, \_C1, \_A1 > &, const [multiset](#)< \_K1, \_C1, \_A1 > &)

#### 4.646.1 Detailed Description

template<typename \_Key, typename \_Compare = std::less<\_Key>, typename \_Alloc = std::allocator<\_Key>> class std::multiset<\_Key, \_Compare, \_Alloc >

A standard container made up of elements, which can be retrieved in logarithmic time.

##### Template Parameters

<code>_Key</code>	Type of key objects.
<code>_Compare</code>	Comparison function object type, defaults to less<_Key>.
<code>_Alloc</code>	Allocator type, defaults to allocator<_Key>.

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using equivalent keys). For a `multiset<Key>` the `key_type` and `value_type` are `Key`.

Multisets support bidirectional iterators.

The private tree data is declared exactly the same way for set and multiset; the distinction is made entirely in how the tree functions are called (\*\_unique versus \*\_equal, same as the standard).

Definition at line 92 of file `stl_multiset.h`.

#### 4.646.2 Constructor & Destructor Documentation

4.646.2.1 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> std::multiset<_Key, _Compare, _Alloc>::multiset( ) [inline]`

Default constructor creates no elements.

Definition at line 137 of file `stl_multiset.h`.

4.646.2.2 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> std::multiset<_Key, _Compare, _Alloc>::multiset( const _Compare & __comp, const allocator_type & __a = allocator_type() ) [inline],[explicit]`

Creates a multiset with no elements.

## Parameters

<code>__comp</code>	Comparator to use.
<code>__a</code>	An allocator object.

Definition at line 146 of file `stl_multiset.h`.

```
4.646.2.3  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            template<typename _InputIterator > std::multiset< _Key, _Compare, _Alloc >::multiset ( _InputIterator __first,
            _InputIterator __last ) [inline]
```

Builds a multiset from a range.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

Create a multiset consisting of copies of the elements from `[first,last)`. This is linear in  $N$  if the range is already sorted, and  $N\log N$  otherwise (where  $N$  is `distance(__first,__last)`).

Definition at line 160 of file `stl_multiset.h`.

```
4.646.2.4  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            template<typename _InputIterator > std::multiset< _Key, _Compare, _Alloc >::multiset ( _InputIterator __first,
            _InputIterator __last, const _Compare & __comp, const allocator_type & __a = allocator_type() )
            [inline]
```

Builds a multiset from a range.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__comp</code>	A comparison functor.
<code>__a</code>	An allocator object.

Create a multiset consisting of copies of the elements from `[__first,__last)`. This is linear in  $N$  if the range is already sorted, and  $N\log N$  otherwise (where  $N$  is `distance(__first,__last)`).

Definition at line 176 of file `stl_multiset.h`.

```
4.646.2.5  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            std::multiset< _Key, _Compare, _Alloc >::multiset ( const multiset< _Key, _Compare, _Alloc > & __x )
            [inline]
```

Multiset copy constructor.

## Parameters

<code>__x</code>	A multiset of identical element and allocator types.
------------------	--

The newly-created multiset uses a copy of the allocation object used by `__x`.

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

```
4.646.2.6  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            std::multiset< _Key, _Compare, _Alloc >::multiset ( multiset< _Key, _Compare, _Alloc > && __x ) [inline],
            [noexcept]
```

Multiset move constructor.

## Parameters

<code>__x</code>	A multiset of identical element and allocator types.
------------------	--

The newly-created multiset contains the exact contents of `__x`. The contents of `__x` are a valid, but unspecified multiset.

Definition at line 200 of file `stl_multiset.h`.

```
4.646.2.7 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset ( initializer_list< value_type > __l, const _Compare & __comp =
_Compare(), const allocator_type & __a = allocator_type() ) [inline]
```

Builds a multiset from an `initializer_list`.

## Parameters

<code>__l</code>	An <code>initializer_list</code> .
<code>__comp</code>	A comparison functor.
<code>__a</code>	An allocator object.

Create a multiset consisting of copies of the elements from the list. This is linear in  $N$  if the list is already sorted, and  $N\log N$  otherwise (where  $N$  is `__l.size()`).

Definition at line 214 of file `stl_multiset.h`.

## 4.646.3 Member Function Documentation

```
4.646.3.1 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator
std::multiset< _Key, _Compare, _Alloc >::begin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 295 of file `stl_multiset.h`.

```
4.646.3.2 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator
std::multiset< _Key, _Compare, _Alloc >::cbegin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 332 of file `stl_multiset.h`.

```
4.646.3.3 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator
std::multiset< _Key, _Compare, _Alloc >::cend ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 341 of file `stl_multiset.h`.

```
4.646.3.4 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> void
std::multiset< _Key, _Compare, _Alloc >::clear ( ) [inline], [noexcept]
```

Erases all elements in a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 616 of file `stl_multiset.h`.

Referenced by `std::multiset< _Key, _Compare, _Alloc >::operator=()`.



4.646.3.5 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
size_type std::multiset<_Key, _Compare, _Alloc>::count( const key_type & __x ) const [inline]`

Finds the number of elements with given key.

## Parameters

<code>__x</code>	Key of elements to be located.
------------------	--------------------------------

## Returns

Number of elements with specified key.

Definition at line 627 of file `stl_multiset.h`.

4.646.3.6 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>  
reverse_iterator std::multiset<_Key, _Compare, _Alloc>::rbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 350 of file `stl_multiset.h`.

4.646.3.7 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>  
reverse_iterator std::multiset<_Key, _Compare, _Alloc>::crend ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 359 of file `stl_multiset.h`.

4.646.3.8 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>  
template<typename... _Args> iterator std::multiset<_Key, _Compare, _Alloc>::emplace ( _Args &&... __args )  
[inline]`

Builds and inserts an element into the multiset.

## Parameters

<code>__args</code>	Arguments used to generate the element instance to be inserted.
---------------------	---

## Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Insertion requires logarithmic time.

Definition at line 409 of file `stl_multiset.h`.

4.646.3.9 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>  
template<typename... _Args> iterator std::multiset<_Key, _Compare, _Alloc>::emplace_hint ( const_iterator __pos,  
_Args &&... __args ) [inline]`

Builds and inserts an element into the multiset.

## Parameters

<code>__pos</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__args</code>	Arguments used to generate the element instance to be inserted.

### Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 435 of file `stl_multiset.h`.

4.646.3.10 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> bool  
std::multiset<_Key, _Compare, _Alloc>::empty ( ) const [inline], [noexcept]`

Returns true if the set is empty.

Definition at line 365 of file `stl_multiset.h`.

4.646.3.11 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator  
std::multiset<_Key, _Compare, _Alloc>::end ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.

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

4.646.3.12 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
std::pair<iterator, iterator> std::multiset<_Key, _Compare, _Alloc>::equal_range ( const key_type & __x )  
[inline]`

Finds a subsequence matching given key.

### Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

### Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 707 of file `stl_multiset.h`.

```
4.646.3.13 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            std::pair<const_iterator, const_iterator> std::multiset<_Key, _Compare, _Alloc >::equal_range ( const key_type
            &__x ) const    [inline]
```

Finds a subsequence matching given key.

**Parameters**

<code>__x</code>	Key to be located.
------------------	--------------------

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 711 of file `stl_multiset.h`.

```
4.646.3.14  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
             _GLIBCXX_ABI_TAG_CXX11 iterator std::multiset< _Key, _Compare, _Alloc >::erase ( const_iterator __position )
             [inline]
```

Erases an element from a multiset.

**Parameters**

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

**Returns**

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 537 of file `stl_multiset.h`.

```
4.646.3.15  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
             size_type std::multiset< _Key, _Compare, _Alloc >::erase ( const key_type & __x ) [inline]
```

Erases elements according to the provided key.

**Parameters**

<code>__x</code>	Key of element to be erased.
------------------	------------------------------

**Returns**

The number of elements erased.

This function erases all elements located by the given key from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 567 of file `stl_multiset.h`.

```
4.646.3.16  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
             _GLIBCXX_ABI_TAG_CXX11 iterator std::multiset< _Key, _Compare, _Alloc >::erase ( const_iterator __first,
             const_iterator __last ) [inline]
```

Erases a [first,last) range of elements from a multiset.

**Parameters**

<code>__first</code>	Iterator pointing to the start of the range to be erased.
<code>__last</code>	Iterator pointing to the end of the range to be erased.

**Returns**

The iterator *last*.

This function erases a sequence of elements from a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 589 of file `stl_multiset.h`.

```
4.646.3.17  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator
            std::multiset<_Key, _Compare, _Alloc>::find ( const key_type & __x ) [inline]
```

Tries to locate an element in a set.

**Parameters**

<code>__x</code>	Element to be located.
------------------	------------------------

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 645 of file `stl_multiset.h`.

```
4.646.3.18  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            const_iterator std::multiset<_Key, _Compare, _Alloc>::find ( const key_type & __x ) const [inline]
```

Tries to locate an element in a set.

**Parameters**

<code>__x</code>	Element to be located.
------------------	------------------------

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 649 of file `stl_multiset.h`.

```
4.646.3.19  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            allocator_type std::multiset<_Key, _Compare, _Alloc>::get_allocator ( ) const [inline], [noexcept]
```

Returns the memory allocation object.

Definition at line 286 of file `stl_multiset.h`.

4.646.3.20 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator  
std::multiset<_Key,_Compare,_Alloc>::insert( const value_type & __x ) [inline]`

Inserts an element into the multiset.



**Parameters**

<code>__x</code>	Element to be inserted.
------------------	-------------------------

**Returns**

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Insertion requires logarithmic time.

Definition at line 454 of file `stl_multiset.h`.

Referenced by `std::multiset< _Key, _Compare, _Alloc >::insert()`, and `std::multiset< _Key, _Compare, _Alloc >::operator=()`.

4.646.3.21 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator std::multiset< _Key, _Compare, _Alloc >::insert ( const_iterator __position, const value_type & __x ) [inline]`

Inserts an element into the multiset.

**Parameters**

<code>__position</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__x</code>	Element to be inserted.

**Returns**

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 484 of file `stl_multiset.h`.

4.646.3.22 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> template<typename _InputIterator > void std::multiset< _Key, _Compare, _Alloc >::insert ( _InputIterator __first, _InputIterator __last ) [inline]`

A template function that tries to insert a range of elements.

**Parameters**

<code>__first</code>	Iterator pointing to the start of the range to be inserted.
<code>__last</code>	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 503 of file `stl_multiset.h`.

```
4.646.3.23  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> void  
            std::multiset<_Key,_Compare,_Alloc>::insert ( initializer_list<value_type> __l )  [inline]
```

Attempts to insert a list of elements into the multiset.

**Parameters**

<code>__l</code>	A <code>std::initializer_list&lt;value_type&gt;</code> of elements to be inserted.
------------------	--

Complexity similar to that of the range constructor.

Definition at line 515 of file `stl_multiset.h`.

References `std::multiset<_Key, _Compare, _Alloc>::insert()`.

4.646.3.24 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
key_compare std::multiset<_Key, _Compare, _Alloc>::key_comp ( ) const [inline]`

Returns the comparison object.

Definition at line 278 of file `stl_multiset.h`.

4.646.3.25 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator  
std::multiset<_Key, _Compare, _Alloc>::lower_bound ( const key_type & __x ) [inline]`

Finds the beginning of a subsequence matching given key.

**Parameters**

<code>__x</code>	Key to be located.
------------------	--------------------

**Returns**

Iterator pointing to first element equal to or greater than key, or `end()`.

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or `end()` if no such element exists.

Definition at line 666 of file `stl_multiset.h`.

4.646.3.26 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
const_iterator std::multiset<_Key, _Compare, _Alloc>::lower_bound ( const key_type & __x ) const [inline]`

Finds the beginning of a subsequence matching given key.

**Parameters**

<code>__x</code>	Key to be located.
------------------	--------------------

**Returns**

Iterator pointing to first element equal to or greater than key, or `end()`.

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or `end()` if no such element exists.

Definition at line 670 of file `stl_multiset.h`.

4.646.3.27 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
size_type std::multiset<_Key, _Compare, _Alloc>::max_size ( ) const [inline], [noexcept]`

Returns the maximum size of the set.

Definition at line 375 of file `stl_multiset.h`.

```
4.646.3.28 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
    multiset& std::multiset< _Key, _Compare, _Alloc >::operator= ( const multiset< _Key, _Compare, _Alloc > & __x
    ) [inline]
```

Multiset assignment operator.

## Parameters

<code>__x</code>	A multiset of identical element and allocator types.
------------------	--

All the elements of `__x` are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 229 of file `stl_multiset.h`.

```
4.646.3.29  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            multiset& std::multiset< _Key, _Compare, _Alloc >::operator= ( multiset< _Key, _Compare, _Alloc > && __x )
            [inline]
```

Multiset move assignment operator.

## Parameters

<code>__x</code>	A multiset of identical element and allocator types.
------------------	--

The contents of `__x` are moved into this multiset (without copying). `__x` is a valid, but unspecified multiset.

Definition at line 245 of file `stl_multiset.h`.

References `std::multiset< _Key, _Compare, _Alloc >::clear()`, and `std::multiset< _Key, _Compare, _Alloc >::swap()`.

```
4.646.3.30  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            multiset& std::multiset< _Key, _Compare, _Alloc >::operator= ( initializer_list< value_type > __l ) [inline]
```

Multiset list assignment operator.

## Parameters

<code>__l</code>	An <code>initializer_list</code> .
------------------	------------------------------------

This function fills a multiset with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the multiset and that the resulting multiset's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 266 of file `stl_multiset.h`.

References `std::multiset< _Key, _Compare, _Alloc >::clear()`, and `std::multiset< _Key, _Compare, _Alloc >::insert()`.

```
4.646.3.31  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            reverse_iterator std::multiset< _Key, _Compare, _Alloc >::rbegin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 313 of file `stl_multiset.h`.

```
4.646.3.32  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            reverse_iterator std::multiset< _Key, _Compare, _Alloc >::rend ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 322 of file `stl_multiset.h`.

```
4.646.3.33  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            size_type std::multiset< _Key, _Compare, _Alloc >::size ( ) const [inline], [noexcept]
```

Returns the size of the set.

Definition at line 370 of file `stl_multiset.h`.

4.646.3.34 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> void  
std::multiset<_Key,_Compare,_Alloc>::swap ( multiset<_Key,_Compare,_Alloc> &__x ) [inline]`

Swaps data with another multiset.

**Parameters**

<code>__x</code>	A multiset of the same element and allocator types.
------------------	---

This exchanges the elements between two multisets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

Definition at line 390 of file `stl_multiset.h`.

Referenced by `std::multiset<_Key, _Compare, _Alloc >::operator=()`, and `std::swap()`.

4.646.3.35 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator  
std::multiset<_Key, _Compare, _Alloc >::upper_bound ( const key_type & __x ) [inline]`

Finds the end of a subsequence matching given key.

**Parameters**

<code>__x</code>	Key to be located.
------------------	--------------------

**Returns**

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 682 of file `stl_multiset.h`.

4.646.3.36 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
const_iterator std::multiset<_Key, _Compare, _Alloc >::upper_bound ( const key_type & __x ) const [inline]`

Finds the end of a subsequence matching given key.

**Parameters**

<code>__x</code>	Key to be located.
------------------	--------------------

**Returns**

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 686 of file `stl_multiset.h`.

4.646.3.37 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
value_compare std::multiset<_Key, _Compare, _Alloc >::value_comp ( ) const [inline]`

Returns the comparison object.

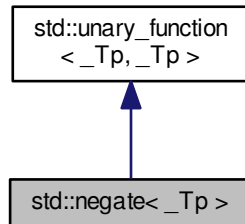
Definition at line 282 of file `stl_multiset.h`.

The documentation for this class was generated from the following file:

- [stl\\_multiset.h](#)

## 4.647 `std::negate<_Tp>` Struct Template Reference

Inheritance diagram for `std::negate<_Tp>`:



### Public Types

- typedef `_Tp` [argument\\_type](#)
- typedef `_Tp` [result\\_type](#)

### Public Member Functions

- `_Tp operator()` (`const _Tp &__x`) `const`

#### 4.647.1 Detailed Description

```
template<typename _Tp>struct std::negate<_Tp>
```

One of the [math functors](#).

Definition at line 185 of file `stl_function.h`.

#### 4.647.2 Member Typedef Documentation

4.647.2.1 typedef `_Tp` `std::unary_function<_Tp, _Tp>::argument_type` `[inherited]`

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

4.647.2.2 typedef `_Tp` `std::unary_function<_Tp, _Tp>::result_type` `[inherited]`

`result_type` is the return type

Definition at line 107 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)



## 4.648 `std::negative_binomial_distribution<_IntType>` Class Template Reference

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_IntType` [result\\_type](#)

### Public Member Functions

- **`negative_binomial_distribution`** (`_IntType` \_\_k=1, double \_\_p=0.5)
- **`negative_binomial_distribution`** (const [param\\_type](#) &\_\_p)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator> void __generate` (`_ForwardIterator` \_\_f, `_ForwardIterator` \_\_t, `_UniformRandomNumberGenerator` &\_\_urng)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator> void __generate` (`_ForwardIterator` \_\_f, `_ForwardIterator` \_\_t, `_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- `template<typename _UniformRandomNumberGenerator> void __generate` ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, `_UniformRandomNumberGenerator` &\_\_urng)
- `template<typename _UniformRandomNumberGenerator> void __generate` ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, `_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- `_IntType k` () const
- [result\\_type max](#) () const
- [result\\_type min](#) () const
- `template<typename _UniformRandomNumberGenerator> result\_type operator\(\)` (`_UniformRandomNumberGenerator` &\_\_urng)
- `template<typename _UniformRandomNumberGenerator> result\_type operator\(\)` (`_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- double [p](#) () const
- [param\\_type param](#) () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()

### Friends

- `template<typename _IntType1, typename _CharT, typename _Traits> std::basic_ostream<_CharT, _Traits> & operator<<` (`std::basic_ostream<_CharT, _Traits>` &\_\_os, const `std::negative_binomial_distribution<_IntType1>` &\_\_x)
- bool [operator==](#) (const `negative_binomial_distribution` &\_\_d1, const `negative_binomial_distribution` &\_\_d2)
- `template<typename _IntType1, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits> & operator>>` (`std::basic_istream<_CharT, _Traits>` &\_\_is, `std::negative_binomial_distribution<_IntType1>` &\_\_x)

#### 4.648.1 Detailed Description

`template<typename _IntType = int> class std::negative_binomial_distribution<_IntType>`

A `negative_binomial_distribution` random number distribution.

The formula for the negative binomial probability mass function is  $p(i) = \binom{n}{i} p^i (1-p)^{t-i}$  where  $t$  and  $p$  are the parameters of the distribution.

Definition at line 4208 of file random.h.

#### 4.648.2 Member Typedef Documentation

4.648.2.1 `template<typename _IntType = int> typedef _IntType std::negative_binomial_distribution<_IntType>::result_type`

The type of the range of the distribution.

Definition at line 4211 of file random.h.

#### 4.648.3 Member Function Documentation

4.648.3.1 `template<typename _IntType = int> _IntType std::negative_binomial_distribution<_IntType>::k ( ) const`  
`[inline]`

Return the  $k$  parameter of the distribution.

Definition at line 4266 of file random.h.

4.648.3.2 `template<typename _IntType = int> result_type std::negative_binomial_distribution<_IntType>::max ( )`  
`const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 4302 of file random.h.

References `std::max()`.

4.648.3.3 `template<typename _IntType = int> result_type std::negative_binomial_distribution<_IntType>::min ( )`  
`const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4295 of file random.h.

4.648.3.4 `template<typename _IntType = int> template<typename UniformRandomNumberGenerator> result_type`  
`std::negative_binomial_distribution<_IntType>::operator() ( UniformRandomNumberGenerator & __urng )`

Generating functions.

4.648.3.5 `template<typename _IntType = int> double std::negative_binomial_distribution<_IntType>::p ( ) const`  
`[inline]`

Return the  $p$  parameter of the distribution.

Definition at line 4273 of file random.h.

4.648.3.6 `template<typename _IntType = int> param_type std::negative_binomial_distribution<_IntType>::param ( )`  
`const [inline]`

Returns the parameter set of the distribution.

Definition at line 4280 of file random.h.

4.648.3.7 `template<typename _IntType = int> void std::negative_binomial_distribution<_IntType>::param ( const`  
`param_type & __param ) [inline]`

Sets the parameter set of the distribution.

## Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 4288 of file random.h.

4.648.3.8 `template<typename _IntType = int> void std::negative_binomial_distribution< _IntType >::reset ( )`  
`[inline]`

Resets the distribution state.

Definition at line 4259 of file random.h.

References `std::gamma_distribution< _RealType >::reset()`.

## 4.648.4 Friends And Related Function Documentation

4.648.4.1 `template<typename _IntType = int> template<typename _IntType1 , typename _CharT , typename _Traits >`  
`std::basic_ostream< _CharT, _Traits>& operator<< ( std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::negative_binomial_distribution< _IntType1 > &__x ) [friend]`

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

## Parameters

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

## Returns

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

4.648.4.2 `template<typename _IntType = int> bool operator== ( const negative_binomial_distribution< _IntType > &__d1,`  
`const negative_binomial_distribution< _IntType > &__d2 ) [friend]`

Return true if two negative binomial distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 4351 of file random.h.

4.648.4.3 `template<typename _IntType = int> template<typename _IntType1 , typename _CharT , typename _Traits`  
`> std::basic_istream< _CharT, _Traits>& operator>> ( std::basic_istream< _CharT, _Traits > &__is,`  
`std::negative_binomial_distribution< _IntType1 > &__x ) [friend]`

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

## Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>negative_binomial_distribution</code> random number generator engine.

## Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

## 4.649 `std::negative_binomial_distribution<_IntType>::param_type` Struct Reference

### Public Types

- typedef `negative_binomial_distribution<_IntType>` `distribution_type`

### Public Member Functions

- `param_type` (`_IntType` `__k`=1, `double` `__p`=0.5)
- `_IntType` `k` () const
- `double` `p` () const

### Friends

- `bool` `operator==` (const `param_type` &`__p1`, const `param_type` &`__p2`)

#### 4.649.1 Detailed Description

`template<typename _IntType = int>struct std::negative_binomial_distribution<_IntType>::param_type`

Parameter type.

Definition at line 4217 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 4.650 `std::nested_exception` Class Reference

Inherited by `std::_Nested_exception<_Except>`.

### Public Member Functions

- `nested_exception` (const `nested_exception` &)=default
- `exception_ptr` `nested_ptr` () const
- `nested_exception` & `operator=` (const `nested_exception` &)=default
- void `rethrow_nested` () const `__attribute__((__noreturn__))`

#### 4.650.1 Detailed Description

Exception class with `exception_ptr` data member.

Definition at line 55 of file `nested_exception.h`.

The documentation for this class was generated from the following file:

- [nested\\_exception.h](#)

## 4.651 `std::normal_distribution<_RealType>` Class Template Reference

### Classes

- struct `param_type`

### Public Types

- typedef `_RealType` `result_type`

### Public Member Functions

- `normal_distribution` (`result_type` \_\_mean=`result_type`(0), `result_type` \_\_stddev=`result_type`(1))
- `normal_distribution` (const `param_type` &\_\_p)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` > void `__generate` (`_ForwardIterator` \_\_f, `_ForwardIterator` \_\_t, `_UniformRandomNumberGenerator` &\_\_urng)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` > void `__generate` (`_ForwardIterator` \_\_f, `_ForwardIterator` \_\_t, `_UniformRandomNumberGenerator` &\_\_urng, const `param_type` &\_\_p)
- template<typename `_UniformRandomNumberGenerator` > void `__generate` (`result_type` \*\_\_f, `result_type` \*\_\_t, `_UniformRandomNumberGenerator` &\_\_urng, const `param_type` &\_\_p)
- `result_type` `max` () const
- `_RealType` `mean` () const
- `result_type` `min` () const
- template<typename `_UniformRandomNumberGenerator` > `result_type` `operator()` (`_UniformRandomNumberGenerator` &\_\_urng)
- template<typename `_UniformRandomNumberGenerator` > `result_type` `operator()` (`_UniformRandomNumberGenerator` &\_\_urng, const `param_type` &\_\_p)
- `param_type` `param` () const
- void `param` (const `param_type` &\_\_param)
- void `reset` ()
- `_RealType` `stddev` () const

### Friends

- template<typename `_RealType1` , typename `_CharT` , typename `_Traits` > `std::basic_ostream<_CharT, _Traits>` & `operator<<` (`std::basic_ostream<_CharT, _Traits>` &\_\_os, const `std::normal_distribution<_RealType1>` &\_\_x)
- template<typename `_RealType1` > bool `operator==` (const `std::normal_distribution<_RealType1>` &\_\_d1, const `std::normal_distribution<_RealType1>` &\_\_d2)
- template<typename `_RealType1` , typename `_CharT` , typename `_Traits` > `std::basic_istream<_CharT, _Traits>` & `operator>>` (`std::basic_istream<_CharT, _Traits>` &\_\_is, `std::normal_distribution<_RealType1>` &\_\_x)

#### 4.651.1 Detailed Description

template<typename `_RealType` = double>class `std::normal_distribution<_RealType>`

A normal continuous distribution for random numbers.

The formula for the normal probability density function is

$$p(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x-\mu}{2\sigma^2}}$$

Definition at line 2085 of file `random.h`.

#### 4.651.2 Member Typedef Documentation

##### 4.651.2.1 `template<typename _RealType = double> typedef _RealType std::normal_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 2088 of file `random.h`.

#### 4.651.3 Constructor & Destructor Documentation

##### 4.651.3.1 `template<typename _RealType = double> std::normal_distribution<_RealType>::normal_distribution ( result_type __mean = result_type(0), result_type __stddev = result_type(1) ) [inline], [explicit]`

Constructs a normal distribution with parameters *mean* and standard deviation.

Definition at line 2130 of file `random.h`.

#### 4.651.4 Member Function Documentation

##### 4.651.4.1 `template<typename _RealType = double> result_type std::normal_distribution<_RealType>::max ( ) const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 2187 of file `random.h`.

##### 4.651.4.2 `template<typename _RealType = double> _RealType std::normal_distribution<_RealType>::mean ( ) const [inline]`

Returns the mean of the distribution.

Definition at line 2151 of file `random.h`.

##### 4.651.4.3 `template<typename _RealType = double> result_type std::normal_distribution<_RealType>::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2180 of file `random.h`.

##### 4.651.4.4 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator> result_type std::normal_distribution<_RealType>::operator() ( _UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 2195 of file `random.h`.

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

##### 4.651.4.5 `template<typename _RealType = double> param_type std::normal_distribution<_RealType>::param ( ) const [inline]`

Returns the parameter set of the distribution.

Definition at line 2165 of file `random.h`.

4.651.4.6 `template<typename _RealType = double> void std::normal_distribution<_RealType>::param ( const param_type & __param ) [inline]`

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 2173 of file random.h.

4.651.4.7 `template<typename _RealType = double> void std::normal_distribution<_RealType>::reset ( ) [inline]`

Resets the distribution state.

Definition at line 2144 of file random.h.

Referenced by `std::lognormal_distribution<_RealType>::reset()`, `std::gamma_distribution<result_type>::reset()`, `std::student_t_distribution<_RealType>::reset()`, `std::binomial_distribution<_IntType>::reset()`, and `std::poisson_distribution<_IntType>::reset()`.

4.651.4.8 `template<typename _RealType = double> _RealType std::normal_distribution<_RealType>::stddev ( ) const [inline]`

Returns the standard deviation of the distribution.

Definition at line 2158 of file random.h.

#### 4.651.5 Friends And Related Function Documentation

4.651.5.1 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits> std::basic_ostream<_CharT, _Traits>& operator<< ( std::basic_ostream<_CharT, _Traits> & __os, const std::normal_distribution<_RealType1> & __x ) [friend]`

Inserts a normal\_distribution random number distribution `__x` into the output stream `__os`.

Parameters

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

Returns

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

4.651.5.2 `template<typename _RealType = double> template<typename _RealType1> bool operator== ( const std::normal_distribution<_RealType1> & __d1, const std::normal_distribution<_RealType1> & __d2 ) [friend]`

Return true if two normal distributions have the same parameters and the sequences that would be generated are equal.

4.651.5.3 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<_CharT, _Traits> & __is, std::normal_distribution<_RealType1> & __x ) [friend]`

Extracts a normal\_distribution random number distribution `__x` from the input stream `__is`.

## Parameters

__is	An input stream.
__x	A normal_distribution random number generator engine.

## Returns

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

The documentation for this class was generated from the following file:

- [random.h](#)

## 4.652 std::normal\_distribution&lt;\_RealType&gt;::param\_type Struct Reference

## Public Types

- typedef [normal\\_distribution](#)<\_RealType> **distribution\_type**

## Public Member Functions

- **param\_type** (\_RealType \_\_mean=\_RealType(0), \_RealType \_\_stddev=\_RealType(1))
- \_RealType **mean** () const
- \_RealType **stddev** () const

## Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

## 4.652.1 Detailed Description

```
template<typename _RealType = double>struct std::normal_distribution<_RealType>::param_type
```

Parameter type.

Definition at line 2094 of file random.h.

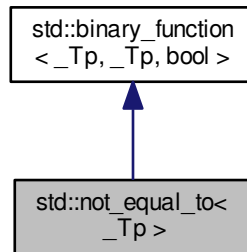
The documentation for this struct was generated from the following file:

- [random.h](#)



#### 4.653 `std::not_equal_to<_Tp>` Struct Template Reference

Inheritance diagram for `std::not_equal_to<_Tp>`:



##### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

##### Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

##### 4.653.1 Detailed Description

`template<typename _Tp>struct std::not_equal_to<_Tp>`

One of the [comparison functors](#).

Definition at line 213 of file `stl_function.h`.

##### 4.653.2 Member Typedef Documentation

4.653.2.1 `typedef _Tp std::binary_function<_Tp, _Tp, bool>::first_argument_type` [\[inherited\]](#)

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.653.2.2 `typedef bool std::binary_function<_Tp, _Tp, bool>::result_type` [\[inherited\]](#)

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.653.2.3 typedef `Tp` `std::binary_function<_Tp, _Tp, bool>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

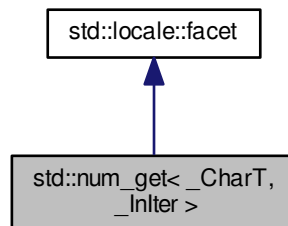
Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.654 std::num\_get< \_CharT, \_InIter > Class Template Reference

Inheritance diagram for `std::num_get< _CharT, _InIter >`:



### Public Types

- typedef `_CharT` [char\\_type](#)
- typedef `_InIter` [iter\\_type](#)

### Public Member Functions

- [num\\_get](#) (`size_t` \_\_refs=0)
- [iter\\_type get](#) (`iter_type` \_\_in, `iter_type` \_\_end, `ios_base &__io`, `ios_base::iostate &__err`, `bool &__v`) const
- [iter\\_type get](#) (`iter_type` \_\_in, `iter_type` \_\_end, `ios_base &__io`, `ios_base::iostate &__err`, `void *&__v`) const
- [iter\\_type get](#) (`iter_type` \_\_in, `iter_type` \_\_end, `ios_base &__io`, `ios_base::iostate &__err`, `long &__v`) const
- [iter\\_type get](#) (`iter_type` \_\_in, `iter_type` \_\_end, `ios_base &__io`, `ios_base::iostate &__err`, `unsigned short &__v`) const
- [iter\\_type get](#) (`iter_type` \_\_in, `iter_type` \_\_end, `ios_base &__io`, `ios_base::iostate &__err`, `unsigned int &__v`) const
- [iter\\_type get](#) (`iter_type` \_\_in, `iter_type` \_\_end, `ios_base &__io`, `ios_base::iostate &__err`, `unsigned long &__v`) const
- [iter\\_type get](#) (`iter_type` \_\_in, `iter_type` \_\_end, `ios_base &__io`, `ios_base::iostate &__err`, `long long &__v`) const
- [iter\\_type get](#) (`iter_type` \_\_in, `iter_type` \_\_end, `ios_base &__io`, `ios_base::iostate &__err`, `unsigned long long &__v`) const
- [iter\\_type get](#) (`iter_type` \_\_in, `iter_type` \_\_end, `ios_base &__io`, `ios_base::iostate &__err`, `float &__v`) const
- [iter\\_type get](#) (`iter_type` \_\_in, `iter_type` \_\_end, `ios_base &__io`, `ios_base::iostate &__err`, `double &__v`) const
- [iter\\_type get](#) (`iter_type` \_\_in, `iter_type` \_\_end, `ios_base &__io`, `ios_base::iostate &__err`, `long double &__v`) const

## Static Public Attributes

- static [locale::id](#) id

## Protected Member Functions

- virtual [~num\\_get](#) ()
- [iter\\_type \\_M\\_extract\\_float](#) (iter\_type, iter\_type, ios\_base &, ios\_base::iostate &, string &) const
- template<typename \_ValueT > [iter\\_type \\_M\\_extract\\_int](#) (iter\_type, iter\_type, ios\_base &, ios\_base::iostate &, \_↔\_ValueT &) const
- template<typename \_CharT2 > [\\_\\_gnu\\_cxx::\\_\\_enable\\_if< \\_\\_is\\_char< \\_CharT2 >::\\_\\_value, int >::\\_\\_type \\_M\\_find](#) (const \_CharT2 \*, size\_t \_\_len, \_CharT2 \_\_c) const
- template<typename \_CharT2 > [\\_\\_gnu\\_cxx::\\_\\_enable\\_if<! \\_\\_is\\_char< \\_CharT2 >::\\_\\_value, int >::\\_\\_type \\_M\\_find](#) (const \_CharT2 \* \_\_zero, size\_t \_\_len, \_CharT2 \_\_c) const
- virtual [iter\\_type do\\_get](#) (iter\_type, iter\_type, ios\_base &, ios\_base::iostate &, bool &) const
- virtual [iter\\_type do\\_get](#) (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, long & \_\_v) const
- virtual [iter\\_type do\\_get](#) (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, unsigned short & \_\_v) const
- virtual [iter\\_type do\\_get](#) (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, unsigned int & \_\_v) const
- virtual [iter\\_type do\\_get](#) (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, unsigned long & \_\_v) const
- virtual [iter\\_type do\\_get](#) (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, long long & \_\_v) const
- virtual [iter\\_type do\\_get](#) (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, unsigned long long & \_\_v) const
- virtual [iter\\_type do\\_get](#) (iter\_type, iter\_type, ios\_base &, ios\_base::iostate &, float &) const
- virtual [iter\\_type do\\_get](#) (iter\_type, iter\_type, ios\_base &, ios\_base::iostate &, double &) const
- virtual [iter\\_type do\\_get](#) (iter\_type, iter\_type, ios\_base &, ios\_base::iostate &, long double &) const
- virtual [iter\\_type do\\_get](#) (iter\_type, iter\_type, ios\_base &, ios\_base::iostate &, void \*&) const

## Static Protected Member Functions

- static [\\_\\_c\\_locale \\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc)
- static [\\_\\_c\\_locale \\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static [\\_\\_c\\_locale \\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \* \_\_s)

### 4.654.1 Detailed Description

```
template<typename _CharT, typename _InIter> class std::num_get< _CharT, _InIter >
```

Primary class template num\_get.

This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators.

The num\_get template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the num\_get facet.

Definition at line 1915 of file locale\_facets.h.

#### 4.654.2 Member Typedef Documentation

##### 4.654.2.1 template<typename \_CharT, typename \_InIter > typedef \_CharT std::num\_get<\_CharT, \_InIter >::char\_type

Public typedefs.

Definition at line 1921 of file locale\_facets.h.

##### 4.654.2.2 template<typename \_CharT, typename \_InIter > typedef \_InIter std::num\_get<\_CharT, \_InIter >::iter\_type

Public typedefs.

Definition at line 1922 of file locale\_facets.h.

#### 4.654.3 Constructor & Destructor Documentation

##### 4.654.3.1 template<typename \_CharT, typename \_InIter > std::num\_get<\_CharT, \_InIter >::num\_get ( size\_t \_\_refs = 0 ) [inline], [explicit]

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

Definition at line 1936 of file locale\_facets.h.

##### 4.654.3.2 template<typename \_CharT, typename \_InIter > virtual std::num\_get<\_CharT, \_InIter >::~~num\_get ( ) [inline], [protected], [virtual]

Destructor.

Definition at line 2108 of file locale\_facets.h.

#### 4.654.4 Member Function Documentation

##### 4.654.4.1 template<typename \_CharT, typename \_InIter > virtual iter\_type std::num\_get<\_CharT, \_InIter >::do\_get ( iter\_type, iter\_type, ios\_base &, ios\_base::iostate &, bool & ) const [protected], [virtual]

Numeric parsing.

Parses the input stream into the variable v. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

**Parameters**

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after reading.

Referenced by `std::num_get<_CharT, _InIter>::get()`.

4.654.4.2 `template<typename _CharT, typename _InIter> virtual iter_type std::num_get<_CharT, _InIter>::do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, long & __v ) const [inline], [protected], [virtual]`

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

**See also**

`get()` for more details.

**Parameters**

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2176 of file `locale_facets.h`.

4.654.4.3 `template<typename _CharT, typename _InIter> virtual iter_type std::num_get<_CharT, _InIter>::do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned short & __v ) const [inline], [protected], [virtual]`

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

**See also**

`get()` for more details.

## Parameters

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

## Returns

Iterator after reading.

Definition at line 2181 of file locale\_facets.h.

```
4.654.4.4 template<typename _CharT, typename _Inlter > virtual iter_type std::num_get<_CharT, _Inlter >::do_get (
    iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned int & __v ) const
    [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

## See also

`get()` for more details.

## Parameters

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

## Returns

Iterator after reading.

Definition at line 2186 of file locale\_facets.h.

```
4.654.4.5 template<typename _CharT, typename _Inlter > virtual iter_type std::num_get<_CharT, _Inlter >::do_get (
    iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long & __v ) const
    [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

## See also

`get()` for more details.

**Parameters**

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2191 of file locale\_facets.h.

```
4.654.4.6 template<typename _CharT, typename _InIter > virtual iter_type std::num_get< _CharT, _InIter >::do_get (
    iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, long long & __v ) const
    [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

**See also**

`get()` for more details.

**Parameters**

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2197 of file locale\_facets.h.

```
4.654.4.7 template<typename _CharT, typename _InIter > virtual iter_type std::num_get< _CharT, _InIter >::do_get (
    iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long long & __v ) const
    [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

**See also**

`get()` for more details.

## Parameters

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

## Returns

Iterator after reading.

Definition at line 2202 of file locale\_facets.h.

4.654.4.8 `template<typename _CharT, typename _Inlter > virtual iter_type std::num_get<_CharT, _Inlter >::do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, float &) const` [protected], [virtual]

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

## See also

`get()` for more details.

## Parameters

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

## Returns

Iterator after reading.

4.654.4.9 `template<typename _CharT, typename _Inlter > virtual iter_type std::num_get<_CharT, _Inlter >::do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, double &) const` [protected], [virtual]

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

## See also

`get()` for more details.

## Parameters

<code>__beg</code>	Start of input stream.
--------------------	------------------------



<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after reading.

4.654.4.10 `template<typename _CharT, typename _InIter > virtual iter_type std::num_get< _CharT, _InIter >::do_get ( iter_type, iter_type, ios_base &, ios_base::iostate &, long double & ) const` [protected], [virtual]

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

**See also**

`get()` for more details.

**Parameters**

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after reading.

4.654.4.11 `template<typename _CharT, typename _InIter > virtual iter_type std::num_get< _CharT, _InIter >::do_get ( iter_type, iter_type, ios_base &, ios_base::iostate &, void *& ) const` [protected], [virtual]

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

**See also**

`get()` for more details.

**Parameters**

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.

<code>__v</code>	Value to format and insert.
------------------	-----------------------------

**Returns**

Iterator after reading.

**4.654.4.12** `template<typename _CharT, typename _InIter > iter_type std::num_get<_CharT, _InIter >::get( iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, bool & __v ) const [inline]`

Numeric parsing.

Parses the input stream into the bool `v`. It does so by calling `num_get::do_get()`.

If `ios_base::boolalpha` is set, attempts to read `ctype<CharT>::truenamename()` or `ctype<CharT>::falsename()`. Sets `v` to true or false if successful. Sets `err` to `ios_base::failbit` if reading the string fails. Sets `err` to `ios_base::eofbit` if the stream is emptied.

If `ios_base::boolalpha` is not set, proceeds as with reading a long, except if the value is 1, sets `v` to true, if the value is 0, sets `v` to false, and otherwise set `err` to `ios_base::failbit`.

**Parameters**

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 1962 of file `locale_facets.h`.

References `std::num_get<_CharT, _InIter >::do_get()`.

**4.654.4.13** `template<typename _CharT, typename _InIter > iter_type std::num_get<_CharT, _InIter >::get( iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, long & __v ) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable `v`. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in `io`.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `numpunct::grouping()` and `numpunct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for `v`, `v` is set. Otherwise, sets `err` to `ios_base::failbit` and leaves `v` unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

**Parameters**

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 1999 of file `locale_facets.h`.

References `std::num_get<_CharT, _InIter>::do_get()`.

4.654.4.14 `template<typename _CharT, typename _InIter> iter_type std::num_get<_CharT, _InIter>::get ( iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned short & __v ) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable `v`. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in `io`.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `numpunct::grouping()` and `numpunct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for `v`, `v` is set. Otherwise, sets `err` to `ios_base::failbit` and leaves `v` unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

**Parameters**

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2004 of file `locale_facets.h`.

References `std::num_get<_CharT, _InIter>::do_get()`.

4.654.4.15 `template<typename _CharT, typename _InIter> iter_type std::num_get<_CharT, _InIter>::get ( iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned int & __v ) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable `v`. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in `io`.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the

scanf o specifier. Else if equal to ios\_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands\_sep(). If the pattern of digit groups isn't consistent, sets err to ios\_base::failbit.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios\_base::failbit and leaves *v* unaltered. Sets err to ios\_base::eofbit if the stream is emptied.

#### Parameters

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

#### Returns

Iterator after reading.

Definition at line 2009 of file locale\_facets.h.

References std::num\_get< \_CharT, \_InIter >::do\_get().

4.654.4.16 `template<typename _CharT, typename _InIter> iter_type std::num_get< _CharT, _InIter >::get( iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long & __v ) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling num\_get::do\_get().

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of io.flags() & ios\_base::basefield. If equal to ios\_base::oct, parses like the scanf o specifier. Else if equal to ios\_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands\_sep(). If the pattern of digit groups isn't consistent, sets err to ios\_base::failbit.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios\_base::failbit and leaves *v* unaltered. Sets err to ios\_base::eofbit if the stream is emptied.

#### Parameters

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

#### Returns

Iterator after reading.

Definition at line 2014 of file locale\_facets.h.

References std::num\_get< \_CharT, \_InIter >::do\_get().

4.654.4.17 `template<typename _CharT, typename _InIter> iter_type std::num_get<_CharT, _InIter>::get ( iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, long long & __v ) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `numpunct::grouping()` and `numpunct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets *err* to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets *err* to `ios_base::failbit` and leaves *v* unaltered. Sets *err* to `ios_base::eofbit` if the stream is emptied.

#### Parameters

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

#### Returns

Iterator after reading.

Definition at line 2020 of file `locale_facets.h`.

References `std::num_get<_CharT, _InIter>::do_get()`.

4.654.4.18 `template<typename _CharT, typename _InIter> iter_type std::num_get<_CharT, _InIter>::get ( iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long long & __v ) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `numpunct::grouping()` and `numpunct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets *err* to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets *err* to `ios_base::failbit` and leaves *v* unaltered. Sets *err* to `ios_base::eofbit` if the stream is emptied.

#### Parameters

<code>__in</code>	Start of input stream.
-------------------	------------------------

<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2025 of file locale\_facets.h.

References `std::num_get< _CharT, _InIter >::do_get()`.

**4.654.4.19** `template<typename _CharT, typename _InIter> iter_type std::num_get< _CharT, _InIter >::get( iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, float & __v ) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable `v`. It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf g` specifier. The matching type length modifier is also used.

The decimal point character used is `num_punct::decimal_point()`. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for `v`, `v` is set. Otherwise, sets `err` to `ios_base::failbit` and leaves `v` unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

**Parameters**

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2059 of file locale\_facets.h.

References `std::num_get< _CharT, _InIter >::do_get()`.

**4.654.4.20** `template<typename _CharT, typename _InIter> iter_type std::num_get< _CharT, _InIter >::get( iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, double & __v ) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable `v`. It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf g` specifier. The matching type length modifier is also used.

The decimal point character used is `num_punct::decimal_point()`. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for `v`, `v` is set. Otherwise, sets `err` to `ios_base::failbit` and leaves `v` unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

**Parameters**

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2064 of file `locale_facets.h`.

References `std::num_get<_CharT, _InIter >::do_get()`.

4.654.4.21 `template<typename _CharT, typename _InIter > iter_type std::num_get<_CharT, _InIter >::get ( iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, long double & __v ) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable `v`. It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf g` specifier. The matching type length modifier is also used.

The decimal point character used is `numpunct::decimal_point()`. Digit grouping is interpreted according to `numpunct::grouping()` and `numpunct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for `v`, `v` is set. Otherwise, sets `err` to `ios_base::failbit` and leaves `v` unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

**Parameters**

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2069 of file `locale_facets.h`.

References `std::num_get<_CharT, _InIter >::do_get()`.

4.654.4.22 `template<typename _CharT, typename _InIter > iter_type std::num_get<_CharT, _InIter >::get ( iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, void *& __v ) const [inline]`

Numeric parsing.

Parses the input stream into the pointer variable `v`. It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf p` specifier.

Digit grouping is interpreted according to `numpunct::grouping()` and `numpunct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

Note that the digit grouping effect for pointers is a bit ambiguous in the standard and shouldn't be relied on. See DR 344.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets *err* to *ios\_base::failbit* and leaves *v* unaltered. Sets *err* to *ios\_base::eofbit* if the stream is emptied.

#### Parameters

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

#### Returns

Iterator after reading.

Definition at line 2102 of file `locale_facets.h`.

References `std::num_get< _CharT, _InIter >::do_get()`.

#### 4.654.5 Member Data Documentation

4.654.5.1 `template<typename _CharT, typename _InIter> locale::id std::num_get< _CharT, _InIter >::id` `[static]`

Numpunct facet id.

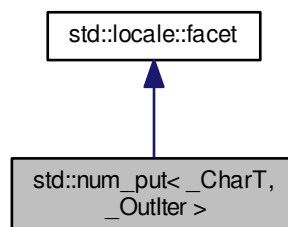
Definition at line 1926 of file `locale_facets.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

#### 4.655 std::num\_put< \_CharT, \_OutIter > Class Template Reference

Inheritance diagram for `std::num_put< _CharT, _OutIter >`:



#### Public Types

- typedef `_CharT` [char\\_type](#)
- typedef `_OutIter` [iter\\_type](#)



## Public Member Functions

- `num_put` (`size_t __refs=0`)
- `iter_type put` (`iter_type __s, ios_base & __io, char_type __fill, bool __v`) `const`
- `iter_type put` (`iter_type __s, ios_base & __io, char_type __fill, const void * __v`) `const`
- `iter_type put` (`iter_type __s, ios_base & __io, char_type __fill, long __v`) `const`
- `iter_type put` (`iter_type __s, ios_base & __io, char_type __fill, unsigned long __v`) `const`
- `iter_type put` (`iter_type __s, ios_base & __io, char_type __fill, long long __v`) `const`
- `iter_type put` (`iter_type __s, ios_base & __io, char_type __fill, unsigned long long __v`) `const`
- `iter_type put` (`iter_type __s, ios_base & __io, char_type __fill, double __v`) `const`
- `iter_type put` (`iter_type __s, ios_base & __io, char_type __fill, long double __v`) `const`

## Static Public Attributes

- static `locale::id id`

## Protected Member Functions

- virtual `~num_put` ()
- void `_M_group_float` (`const char * __grouping, size_t __grouping_size, char_type __sep, const char_type * __p, char_type * __new, char_type * __cs, int & __len`) `const`
- void `_M_group_int` (`const char * __grouping, size_t __grouping_size, char_type __sep, ios_base & __io, char_type * __new, char_type * __cs, int & __len`) `const`
- template<typename \_ValueT > `iter_type _M_insert_float` (`iter_type, ios_base & __io, char_type __fill, char __mod, _ValueT __v`) `const`
- template<typename \_ValueT > `iter_type _M_insert_int` (`iter_type, ios_base & __io, char_type __fill, _ValueT __v`) `const`
- void `_M_pad` (`char_type __fill, streamsize __w, ios_base & __io, char_type * __new, const char_type * __cs, int & __len`) `const`
- virtual `iter_type do_put` (`iter_type __s, ios_base & __io, char_type __fill, bool __v`) `const`
- virtual `iter_type do_put` (`iter_type __s, ios_base & __io, char_type __fill, long __v`) `const`
- virtual `iter_type do_put` (`iter_type __s, ios_base & __io, char_type __fill, unsigned long __v`) `const`
- virtual `iter_type do_put` (`iter_type __s, ios_base & __io, char_type __fill, long long __v`) `const`
- virtual `iter_type do_put` (`iter_type __s, ios_base & __io, char_type __fill, unsigned long long __v`) `const`
- virtual `iter_type do_put` (`iter_type, ios_base & __io, char_type __fill, double`) `const`
- virtual `iter_type do_put` (`iter_type, ios_base & __io, char_type __fill, long double`) `const`
- virtual `iter_type do_put` (`iter_type, ios_base & __io, char_type __fill, const void *`) `const`

## Static Protected Member Functions

- static `__c_locale _S_clone_c_locale` (`__c_locale & __cloc`) `throw ()`
- static void `_S_create_c_locale` (`__c_locale & __cloc, const char * __s, __c_locale __old=0`)
- static void `_S_destroy_c_locale` (`__c_locale & __cloc`)
- static `__c_locale _S_get_c_locale` ()
- static const char \* `_S_get_c_name` () `throw ()`
- static `__c_locale _S_lc_ctype_c_locale` (`__c_locale __cloc, const char * __s`)

## 4.655.1 Detailed Description

```
template<typename _CharT, typename _Outiter>class std::num_put<_CharT, _Outiter>
```

Primary class template num\_put.

This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.

The num\_put template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the num\_put facet.

Definition at line 2254 of file locale\_facets.h.

## 4.655.2 Member Typedef Documentation

```
4.655.2.1 template<typename _CharT, typename _Outiter> typedef _CharT std::num_put<_CharT, _Outiter>::char_type
```

Public typedefs.

Definition at line 2260 of file locale\_facets.h.

```
4.655.2.2 template<typename _CharT, typename _Outiter> typedef _Outiter std::num_put<_CharT, _Outiter>::iter_type
```

Public typedefs.

Definition at line 2261 of file locale\_facets.h.

## 4.655.3 Constructor &amp; Destructor Documentation

```
4.655.3.1 template<typename _CharT, typename _Outiter> std::num_put<_CharT, _Outiter>::num_put ( size_t __refs = 0
    ) [inline],[explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

Definition at line 2275 of file locale\_facets.h.

```
4.655.3.2 template<typename _CharT, typename _Outiter> virtual std::num_put<_CharT, _Outiter>::~~num_put ( )
    [inline],[protected],[virtual]
```

Destructor.

Definition at line 2454 of file locale\_facets.h.

## 4.655.4 Member Function Documentation

```
4.655.4.1 template<typename _CharT, typename _Outiter> virtual iter_type std::num_put<_CharT, _Outiter>::do_put (
    iter_type __s, ios_base& __io, char_type __fill, bool __v ) const [protected],[virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

## Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

## Returns

Iterator after writing.

Referenced by `std::num_put< _CharT, _Outlter >::put()`.

```
4.655.4.2 template<typename _CharT, typename _Outlter > virtual iter_type std::num_put< _CharT, _Outlter >::do_put
( iter_type __s, ios_base & __io, char_type __fill, long __v ) const [inline], [protected],
[virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

## Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

## Returns

Iterator after writing.

Definition at line 2474 of file `locale_facets.h`.

```
4.655.4.3 template<typename _CharT, typename _Outlter > virtual iter_type std::num_put< _CharT, _Outlter >::do_put (
iter_type __s, ios_base & __io, char_type __fill, unsigned long __v ) const [inline], [protected],
[virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

## Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

## Returns

Iterator after writing.

Definition at line 2478 of file `locale_facets.h`.

4.655.4.4 `template<typename _CharT, typename _Outiter > virtual iter_type std::num_put<_CharT, _Outiter >::do_put ( iter_type __s, ios_base & __io, char_type __fill, long long __v ) const [inline], [protected], [virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

#### Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	<code>Char_type</code> to use for filling.
<code>__v</code>	Value to format and insert.

#### Returns

Iterator after writing.

Definition at line 2484 of file `locale_facets.h`.

4.655.4.5 `template<typename _CharT, typename _Outiter > virtual iter_type std::num_put<_CharT, _Outiter >::do_put ( iter_type __s, ios_base & __io, char_type __fill, unsigned long long __v ) const [inline], [protected], [virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

#### Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	<code>Char_type</code> to use for filling.
<code>__v</code>	Value to format and insert.

#### Returns

Iterator after writing.

Definition at line 2489 of file `locale_facets.h`.

4.655.4.6 `template<typename _CharT, typename _Outiter > virtual iter_type std::num_put<_CharT, _Outiter >::do_put ( iter_type, ios_base &, char_type, double ) const [protected], [virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

#### Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after writing.

4.655.4.7 `template<typename _CharT, typename _Outiter> virtual iter_type std::num_put<_CharT, _Outiter>::do_put (iter_type, ios_base &, char_type, long double) const` `[protected]`, `[virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

**Parameters**

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after writing.

4.655.4.8 `template<typename _CharT, typename _Outiter> virtual iter_type std::num_put<_CharT, _Outiter>::do_put (iter_type, ios_base &, char_type, const void *) const` `[protected]`, `[virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

**Parameters**

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after writing.

4.655.4.9 `template<typename _CharT, typename _Outiter> iter_type std::num_put<_CharT, _Outiter>::put (iter_type __s, ios_base & __io, char_type __fill, bool __v) const` `[inline]`

Numeric formatting.

Formats the boolean `v` and inserts it into a stream. It does so by calling `num_put::do_put()`.

If `ios_base::boolalpha` is set, writes `ctype<CharT>::truename()` or `ctype<CharT>::falsename()`. Otherwise formats `v` as an int.

**Parameters**

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after writing.

Definition at line 2293 of file locale\_facets.h.

References `std::num_put<_CharT, _Outlter>::do_put()`.

**4.655.4.10** `template<typename _CharT, typename _Outlter> iter_type std::num_put<_CharT, _Outlter>::put ( iter_type __s, ios_base & __io, char_type __fill, long __v ) const [inline]`

Numeric formatting.

Formats the integral value `v` and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in `io`.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf` `o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showbase` is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

The decimal point character used is `num_punct::decimal_point()`. Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`.

If `io.width()` is non-zero, enough `fill` characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

**Parameters**

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after writing.

Definition at line 2335 of file locale\_facets.h.

References `std::num_put<_CharT, _Outlter>::do_put()`.

**4.655.4.11** `template<typename _CharT, typename _Outlter> iter_type std::num_put<_CharT, _Outlter>::put ( iter_type __s, ios_base & __io, char_type __fill, unsigned long __v ) const [inline]`

Numeric formatting.

Formats the integral value `v` and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, `'+'` is output before positive values. If `ios_base::showbase` is set, `'0'` precedes octal values (except 0) and `'0[xX]'` precedes hex values.

The decimal point character used is `numput::decimal_point()`. Thousands separators are inserted according to `numput::grouping()` and `numput::thousands_sep()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a `'+'` or `'-'` or after `'0x'` or `'0X'`. Otherwise, padding occurs at the beginning.

#### Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	<code>Char_type</code> to use for filling.
<code>__v</code>	Value to format and insert.

#### Returns

Iterator after writing.

Definition at line 2339 of file `locale_facets.h`.

References `std::num_put< _CharT, _Outlter >::do_put()`.

**4.655.4.12** `template<typename _CharT, typename _Outlter> iter_type std::num_put< _CharT, _Outlter >::put ( iter_type __s, ios_base & __io, char_type __fill, long long __v ) const [inline]`

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, `'+'` is output before positive values. If `ios_base::showbase` is set, `'0'` precedes octal values (except 0) and `'0[xX]'` precedes hex values.

The decimal point character used is `numput::decimal_point()`. Thousands separators are inserted according to `numput::grouping()` and `numput::thousands_sep()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a `'+'` or `'-'` or after `'0x'` or `'0X'`. Otherwise, padding occurs at the beginning.

#### Parameters



<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after writing.

Definition at line 2345 of file locale\_facets.h.

References `std::num_put<_CharT, _Outlter>::do_put()`.

**4.655.4.13** `template<typename _CharT, typename _Outlter> iter_type std::num_put<_CharT, _Outlter>::put ( iter_type __s, ios_base & __io, char_type __fill, unsigned long long __v ) const [inline]`

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the printf `o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showbase` is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

The decimal point character used is `numput::decimal_point()`. Thousands separators are inserted according to `numput::grouping()` and `numput::thousands_sep()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

**Parameters**

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after writing.

Definition at line 2349 of file locale\_facets.h.

References `std::num_put<_CharT, _Outlter>::do_put()`.

**4.655.4.14** `template<typename _CharT, typename _Outlter> iter_type std::num_put<_CharT, _Outlter>::put ( iter_type __s, ios_base & __io, char_type __fill, double __v ) const [inline]`

Numeric formatting.

Formats the floating point value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::floatfield`. If equal to `ios_base::fixed`, formats like the `printf f` specifier. Else if equal to `ios_base::scientific`, formats like `e` or `E` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `g` or `G` depending on uppercase. Note that if both fixed and scientific are set, the effect will also be like `g` or `G`.

The output precision is given by `io.precision()`. This precision is capped at `numeric_limits::digits10 + 2` (different for double and long double). The default precision is 6.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showpoint` is set, a decimal point will always be output.

The decimal point character used is `numput::decimal_point()`. Thousands separators are inserted according to `numput::grouping()` and `numput::thousands_sep()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

#### Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

#### Returns

Iterator after writing.

Definition at line 2398 of file `locale_facets.h`.

References `std::num_put< _CharT, _OutIter >::do_put()`.

**4.655.4.15** `template<typename _CharT, typename _OutIter> iter_type std::num_put< _CharT, _OutIter >::put ( iter_type __s, ios_base & __io, char_type __fill, long double __v ) const [inline]`

Numeric formatting.

Formats the floating point value `v` and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in `io`.

The basic format is affected by the value of `io.flags()` & `ios_base::floatfield`. If equal to `ios_base::fixed`, formats like the `printf f` specifier. Else if equal to `ios_base::scientific`, formats like `e` or `E` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `g` or `G` depending on uppercase. Note that if both fixed and scientific are set, the effect will also be like `g` or `G`.

The output precision is given by `io.precision()`. This precision is capped at `numeric_limits::digits10 + 2` (different for double and long double). The default precision is 6.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showpoint` is set, a decimal point will always be output.

The decimal point character used is `numput::decimal_point()`. Thousands separators are inserted according to `numput::grouping()` and `numput::thousands_sep()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

**Parameters**

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after writing.

Definition at line 2402 of file locale\_facets.h.

References `std::num_put<_CharT, _Outlter>::do_put()`.

4.655.4.16 `template<typename _CharT, typename _Outlter> iter_type std::num_put<_CharT, _Outlter>::put ( iter_type __s, ios_base & __io, char_type __fill, const void * __v ) const [inline]`

Numeric formatting.

Formats the pointer value `v` and inserts it into a stream. It does so by calling `num_put::do_put()`.

This function formats `v` as an unsigned long with `ios_base::hex` and `ios_base::showbase` set.

**Parameters**

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after writing.

Definition at line 2423 of file locale\_facets.h.

References `std::num_put<_CharT, _Outlter>::do_put()`.

**4.655.5 Member Data Documentation**

4.655.5.1 `template<typename _CharT, typename _Outlter> locale::id std::num_put<_CharT, _Outlter>::id [static]`

Numpunct facet id.

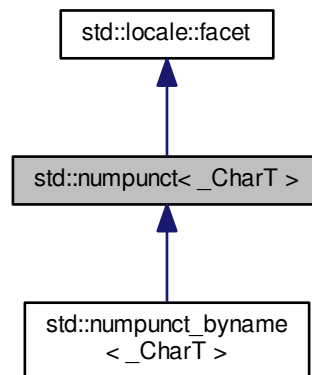
Definition at line 2265 of file locale\_facets.h.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 4.656 std::num\_punct&lt;\_CharT&gt; Class Template Reference

Inheritance diagram for std::num\_punct<\_CharT>:



## Public Types

- typedef \_\_num\_punct\_cache<\_CharT> \_\_cache\_type
- typedef \_CharT char\_type
- typedef basic\_string<\_CharT> string\_type

## Public Member Functions

- num\_punct (size\_t \_\_refs=0)
- num\_punct (\_\_cache\_type \*\_\_cache, size\_t \_\_refs=0)
- num\_punct (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- char\_type decimal\_point () const
- string\_type falsename () const
- string grouping () const
- char\_type thousands\_sep () const
- string\_type truename () const

## Static Public Attributes

- static locale::id id

## Protected Member Functions

- virtual ~num\_punct ()
- void \_M\_initialize\_num\_punct (\_\_c\_locale \_\_cloc=0)

- `template<> void _M_initialize_numpunct (__c_locale __cloc)`
- `template<> void _M_initialize_numpunct (__c_locale __cloc)`
- virtual `char_type do_decimal_point () const`
- virtual `string_type do_falsename () const`
- virtual `string do_grouping () const`
- virtual `char_type do_thousands_sep () const`
- virtual `string_type do_truename () const`

#### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static `void _S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static `void _S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

#### Protected Attributes

- `__cache_type * _M_data`

#### 4.656.1 Detailed Description

`template<typename _CharT>class std::numpunct< _CharT >`

Primary class template `numpunct`.

This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The `numpunct` facet is used by streams for many I/O operations involving numbers.

The `numpunct` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from a `numpunct` facet.

Definition at line 1641 of file `locale_facets.h`.

#### 4.656.2 Member Typedef Documentation

4.656.2.1 `template<typename _CharT > typedef _CharT std::numpunct< _CharT >::char_type`

Public typedefs.

Definition at line 1647 of file `locale_facets.h`.

4.656.2.2 `template<typename _CharT > typedef basic_string<_CharT> std::numpunct< _CharT >::string_type`

Public typedefs.

Definition at line 1648 of file `locale_facets.h`.

## 4.656.3 Constructor &amp; Destructor Documentation

4.656.3.1 `template<typename _CharT> std::numpunct<_CharT>::numpunct ( size_t __refs = 0 ) [inline],  
[explicit]`

Numpunct constructor.

## Parameters

<code>__refs</code>	RefCount to pass to the base class.
---------------------	-------------------------------------

Definition at line 1665 of file locale\_facets.h.

```
4.656.3.2  template<typename _CharT> std::numpunct<_CharT>::numpunct ( __cache_type * __cache, size_t __refs = 0
    ) [inline],[explicit]
```

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up the predefined locale facets.

## Parameters

<code>__cache</code>	<code>__numpunct_cache</code> object.
<code>__refs</code>	RefCount to pass to the base class.

Definition at line 1679 of file locale\_facets.h.

```
4.656.3.3  template<typename _CharT> std::numpunct<_CharT>::numpunct ( __c_locale __cloc, size_t __refs = 0 )
    [inline],[explicit]
```

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

## Parameters

<code>__cloc</code>	The C locale.
<code>__refs</code>	RefCount to pass to the base class.

Definition at line 1693 of file locale\_facets.h.

```
4.656.3.4  template<typename _CharT> virtual std::numpunct<_CharT>::~numpunct ( ) [protected],
    [virtual]
```

Destructor.

## 4.656.4 Member Function Documentation

```
4.656.4.1  template<typename _CharT> char_type std::numpunct<_CharT>::decimal_point ( ) const [inline]
```

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning `numpunct<char_type>::do_decimal_point()`.

## Returns

*char\_type* representing a decimal point.

Definition at line 1707 of file locale\_facets.h.

References `std::numpunct<_CharT>::do_decimal_point()`.

```
4.656.4.2  template<typename _CharT> virtual char_type std::numpunct<_CharT>::do_decimal_point ( ) const
    [inline],[protected],[virtual]
```

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

#### Returns

*char\_type* representing a decimal point.

Definition at line 1794 of file `locale_facets.h`.

Referenced by `std::num_punct<_CharT>::decimal_point()`.

**4.656.4.3** `template<typename _CharT> virtual string_type std::num_punct<_CharT>::do_falsename ( ) const`  
[inline], [protected], [virtual]

Return string representation of bool false.

Returns a `string_type` containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.

#### Returns

`string_type` representing printed form of false.

Definition at line 1845 of file `locale_facets.h`.

Referenced by `std::num_punct<_CharT>::falsename()`.

**4.656.4.4** `template<typename _CharT> virtual string std::num_punct<_CharT>::do_grouping ( ) const` [inline],  
[protected], [virtual]

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

#### See also

`grouping()` for details.

#### Returns

String representing grouping specification.

Definition at line 1819 of file `locale_facets.h`.

Referenced by `std::num_punct<_CharT>::grouping()`.

**4.656.4.5** `template<typename _CharT> virtual char_type std::num_punct<_CharT>::do_thousands_sep ( ) const`  
[inline], [protected], [virtual]

Return thousands separator character.

Returns a `char_type` to use as a thousands separator. This function is a hook for derived classes to change the value returned.

#### Returns

*char\_type* representing a thousands separator.

Definition at line 1806 of file `locale_facets.h`.

Referenced by `std::num_punct<_CharT>::thousands_sep()`.



**4.656.4.6** `template<typename _CharT> virtual string_type std::numpunct<_CharT>::do_truename ( ) const`  
`[inline], [protected], [virtual]`

Return string representation of bool true.

Returns a string\_type containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

#### Returns

string\_type representing printed form of true.

Definition at line 1832 of file locale\_facets.h.

Referenced by std::numpunct<\_CharT>::truename().

**4.656.4.7** `template<typename _CharT> string_type std::numpunct<_CharT>::falsename ( ) const` `[inline]`

Return string representation of bool false.

This function returns a string\_type containing the text representation for false bool variables. It does so by calling numpunct<char\_type>::do\_falsename().

#### Returns

string\_type representing printed form of false.

Definition at line 1777 of file locale\_facets.h.

References std::numpunct<\_CharT>::do\_falsename().

**4.656.4.8** `template<typename _CharT> string std::numpunct<_CharT>::grouping ( ) const` `[inline]`

Return grouping specification.

This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the grouping() returns "\003\002" and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was "32", this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling numpunct<char\_type>::do\_grouping().

#### Returns

string representing grouping specification.

Definition at line 1751 of file locale\_facets.h.

References std::numpunct<\_CharT>::do\_grouping().

**4.656.4.9** `template<typename _CharT> char_type std::numpunct<_CharT>::thousands_sep ( ) const` `[inline]`

Return thousands separator character.

This function returns a char\_type to use as a thousands separator. It does so by returning returning numpunct<char\_type>::do\_thousands\_sep().

**Returns**

char\_type representing a thousands separator.

Definition at line 1720 of file locale\_facets.h.

References std::numpunct<\_CharT>::do\_thousands\_sep().

**4.656.4.10** template<typename \_CharT> string\_type std::numpunct<\_CharT>::truename ( ) const [inline]

Return string representation of bool true.

This function returns a string\_type containing the text representation for true bool variables. It does so by calling numpunct<char\_type>::do\_truename().

**Returns**

string\_type representing printed form of true.

Definition at line 1764 of file locale\_facets.h.

References std::numpunct<\_CharT>::do\_truename().

**4.656.5 Member Data Documentation**

**4.656.5.1** template<typename \_CharT> locale::id std::numpunct<\_CharT>::id [static]

Numpunct facet id.

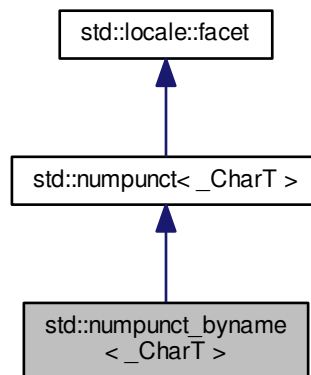
Definition at line 1657 of file locale\_facets.h.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

#### 4.657 `std::numpunct_byname<_CharT>` Class Template Reference

Inheritance diagram for `std::numpunct_byname<_CharT>`:



##### Public Types

- typedef `__numpunct_cache<_CharT>` **`__cache_type`**
- typedef `_CharT` **`char_type`**
- typedef `basic_string<_CharT>` **`string_type`**

##### Public Member Functions

- **`numpunct_byname`** (`const char *__s`, `size_t __refs=0`)
- `char_type decimal_point` () const
- `string_type falsename` () const
- `string grouping` () const
- `char_type thousands_sep` () const
- `string_type truename` () const

##### Static Public Attributes

- static `locale::id` `id`

##### Protected Member Functions

- void **`_M_initialize_numpunct`** (`__c_locale __cloc=0`)
- template<> void **`_M_initialize_numpunct`** (`__c_locale __cloc`)
- template<> void **`_M_initialize_numpunct`** (`__c_locale __cloc`)
- virtual `char_type do_decimal_point` () const
- virtual `string_type do_falsename` () const

- virtual [string do\\_grouping](#) () const
- virtual [char\\_type do\\_thousands\\_sep](#) () const
- virtual [string\\_type do\\_truename](#) () const

#### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

#### Protected Attributes

- `__cache_type * _M_data`

#### 4.657.1 Detailed Description

template<typename \_CharT>class std::numpunct\_byname<\_CharT>

class numpunct\_byname [22.2.3.2].

Definition at line 1874 of file locale\_facets.h.

#### 4.657.2 Member Function Documentation

4.657.2.1 template<typename \_CharT> char\_type std::numpunct<\_CharT>::decimal\_point ( ) const [inline], [inherited]

Return decimal point character.

This function returns a char\_type to use as a decimal point. It does so by returning returning numpunct<char\_type>::do\_decimal\_point().

#### Returns

*char\_type* representing a decimal point.

Definition at line 1707 of file locale\_facets.h.

References std::numpunct<\_CharT>::do\_decimal\_point().

4.657.2.2 template<typename \_CharT> virtual char\_type std::numpunct<\_CharT>::do\_decimal\_point ( ) const [inline], [protected], [virtual], [inherited]

Return decimal point character.

Returns a char\_type to use as a decimal point. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a decimal point.

Definition at line 1794 of file locale\_facets.h.

Referenced by `std::numpunct<_CharT>::decimal_point()`.

**4.657.2.3** `template<typename _CharT> virtual string_type std::numpunct<_CharT>::do_falsename ( ) const`  
`[inline], [protected], [virtual], [inherited]`

Return string representation of bool false.

Returns a *string\_type* containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.

**Returns**

*string\_type* representing printed form of false.

Definition at line 1845 of file locale\_facets.h.

Referenced by `std::numpunct<_CharT>::falsename()`.

**4.657.2.4** `template<typename _CharT> virtual string std::numpunct<_CharT>::do_grouping ( ) const` `[inline],`  
`[protected], [virtual], [inherited]`

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

**See also**

`grouping()` for details.

**Returns**

String representing grouping specification.

Definition at line 1819 of file locale\_facets.h.

Referenced by `std::numpunct<_CharT>::grouping()`.

**4.657.2.5** `template<typename _CharT> virtual char_type std::numpunct<_CharT>::do_thousands_sep ( ) const`  
`[inline], [protected], [virtual], [inherited]`

Return thousands separator character.

Returns a *char\_type* to use as a thousands separator. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a thousands separator.

Definition at line 1806 of file locale\_facets.h.

Referenced by `std::numpunct<_CharT>::thousands_sep()`.

4.657.2.6 `template<typename _CharT> virtual string_type std::num_punct<_CharT>::do_truename ( ) const`  
`[inline], [protected], [virtual], [inherited]`

Return string representation of bool true.

Returns a string\_type containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

#### Returns

string\_type representing printed form of true.

Definition at line 1832 of file locale\_facets.h.

Referenced by std::num\_punct<\_CharT>::truename().

4.657.2.7 `template<typename _CharT> string_type std::num_punct<_CharT>::falsename ( ) const` `[inline],`  
`[inherited]`

Return string representation of bool false.

This function returns a string\_type containing the text representation for false bool variables. It does so by calling num\_punct<char\_type>::do\_falsename().

#### Returns

string\_type representing printed form of false.

Definition at line 1777 of file locale\_facets.h.

References std::num\_punct<\_CharT>::do\_falsename().

4.657.2.8 `template<typename _CharT> string std::num_punct<_CharT>::grouping ( ) const` `[inline],`  
`[inherited]`

Return grouping specification.

This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the grouping() returns "\003\002" and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was "32", this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling num\_punct<char\_type>::do\_grouping().

#### Returns

string representing grouping specification.

Definition at line 1751 of file locale\_facets.h.

References std::num\_punct<\_CharT>::do\_grouping().

**4.657.2.9** `template<typename _CharT> char_type std::numpunct<_CharT>::thousands_sep ( ) const [inline],  
[inherited]`

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning `numpunct<char_type>::do_thousands_sep()`.

#### Returns

`char_type` representing a thousands separator.

Definition at line 1720 of file `locale_facets.h`.

References `std::numpunct<_CharT>::do_thousands_sep()`.

**4.657.2.10** `template<typename _CharT> string_type std::numpunct<_CharT>::truename ( ) const [inline],  
[inherited]`

Return string representation of `bool true`.

This function returns a `string_type` containing the text representation for true bool variables. It does so by calling `numpunct<char_type>::do_truename()`.

#### Returns

`string_type` representing printed form of `true`.

Definition at line 1764 of file `locale_facets.h`.

References `std::numpunct<_CharT>::do_truename()`.

### 4.657.3 Member Data Documentation

**4.657.3.1** `template<typename _CharT> locale::id std::numpunct<_CharT>::id [static], [inherited]`

Numpunct facet id.

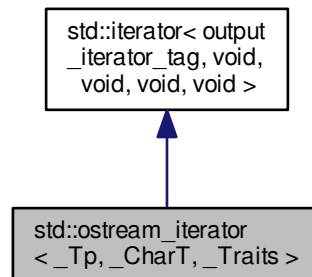
Definition at line 1657 of file `locale_facets.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

4.658 `std::ostream_iterator< _Tp, _CharT, _Traits >` Class Template Reference

Inheritance diagram for `std::ostream_iterator< _Tp, _CharT, _Traits >`:



## Public Types

- typedef void [difference\\_type](#)
- typedef [output\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value\\_type](#)
- typedef [\\_CharT](#) [char\\_type](#)
- typedef [\\_Traits](#) [traits\\_type](#)
- typedef `basic_ostream< _CharT, _Traits >` [ostream\\_type](#)

## Public Member Functions

- [ostream\\_iterator](#) ([ostream\\_type](#) &\_\_s)
- [ostream\\_iterator](#) ([ostream\\_type](#) &\_\_s, const [\\_CharT](#) \*\_\_c)
- [ostream\\_iterator](#) (const [ostream\\_iterator](#) &\_\_obj)
- [ostream\\_iterator](#) & **operator\*** ()
- [ostream\\_iterator](#) & **operator++** ()
- [ostream\\_iterator](#) & **operator++** (int)
- [ostream\\_iterator](#) & **operator=** (const [\\_Tp](#) &\_\_value)

## 4.658.1 Detailed Description

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>class std::ostream_iterator< _Tp, _CharT, _Traits >
```

Provides output iterator semantics for streams.

This class provides an iterator to write to an ostream. The type `Tp` is the only type written by this iterator and there must be an `operator<<(Tp)` defined.



## Template Parameters

<code>_Tp</code>	The type to write to the ostream.
<code>_CharT</code>	The ostream char_type.
<code>_Traits</code>	The ostream char_traits.

Definition at line 154 of file stream\_iterator.h.

## 4.658.2 Member Typedef Documentation

4.658.2.1 `template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>> typedef _CharT  
std::ostream_iterator< _Tp, _CharT, _Traits >::char_type`

Public typedef.

Definition at line 160 of file stream\_iterator.h.

4.658.2.2 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::difference_type [inherited]`

Distance between iterators is represented as this type.

Definition at line 125 of file stl\_iterator\_base\_types.h.

4.658.2.3 `typedef output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >::iterator_category  
[inherited]`

One of the [tag types](#).

Definition at line 121 of file stl\_iterator\_base\_types.h.

4.658.2.4 `template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>> typedef  
basic_ostream<_CharT, _Traits> std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_type`

Public typedef.

Definition at line 162 of file stream\_iterator.h.

4.658.2.5 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer [inherited]`

This type represents a pointer-to-value\_type.

Definition at line 127 of file stl\_iterator\_base\_types.h.

4.658.2.6 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference [inherited]`

This type represents a reference-to-value\_type.

Definition at line 129 of file stl\_iterator\_base\_types.h.

4.658.2.7 `template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>> typedef _Traits  
std::ostream_iterator< _Tp, _CharT, _Traits >::traits_type`

Public typedef.

Definition at line 161 of file stream\_iterator.h.

4.658.2.8 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 123 of file `stl_iterator_base_types.h`.

#### 4.658.3 Constructor & Destructor Documentation

4.658.3.1 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>  
std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator ( ostream_type & __s ) [inline]`

Construct from an ostream.

Definition at line 171 of file `stream_iterator.h`.

4.658.3.2 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>  
std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator ( ostream_type & __s, const _CharT * __c )  
[inline]`

Construct from an ostream.

The delimiter string *c* is written to the stream after every *Tp* written to the stream. The delimiter is not copied, and thus must not be destroyed while this iterator is in use.

##### Parameters

<code>__s</code>	Underlying ostream to write to.
<code>__c</code>	<i>CharT</i> delimiter string to insert.

Definition at line 183 of file `stream_iterator.h`.

4.658.3.3 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>  
std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator ( const ostream_iterator< _Tp, _CharT,  
_Traits > & __obj ) [inline]`

Copy constructor.

Definition at line 187 of file `stream_iterator.h`.

#### 4.658.4 Member Function Documentation

4.658.4.1 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>> ostream_iterator&  
std::ostream_iterator< _Tp, _CharT, _Traits >::operator= ( const _Tp & __value ) [inline]`

Writes *value* to underlying ostream using `operator<<`. If constructed with delimiter string, writes delimiter to ostream.

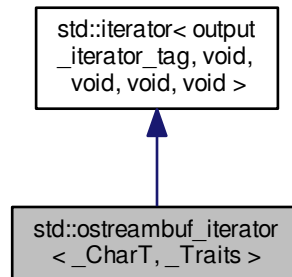
Definition at line 193 of file `stream_iterator.h`.

The documentation for this class was generated from the following file:

- [stream\\_iterator.h](#)

#### 4.659 `std::ostreambuf_iterator<_CharT, _Traits>` Class Template Reference

Inheritance diagram for `std::ostreambuf_iterator<_CharT, _Traits>`:



#### Public Types

- typedef void [difference\\_type](#)
- typedef [output\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value\\_type](#)
  
- typedef [\\_CharT](#) [char\\_type](#)
- typedef [\\_Traits](#) [traits\\_type](#)
- typedef [basic\\_streambuf<\\_CharT, \\_Traits>](#) [streambuf\\_type](#)
- typedef [basic\\_ostream<\\_CharT, \\_Traits>](#) [ostream\\_type](#)

#### Public Member Functions

- [ostreambuf\\_iterator](#) ([ostream\\_type](#) &\_\_s) noexcept
- [ostreambuf\\_iterator](#) ([streambuf\\_type](#) \*\_\_s) noexcept
- [ostreambuf\\_iterator](#) & [M\\_put](#) (const [\\_CharT](#) \*\_\_ws, [streamsize](#) \_\_len)
- bool [failed](#) () const noexcept
- [ostreambuf\\_iterator](#) & [operator\\*](#) ()
- [ostreambuf\\_iterator](#) & [operator++](#) (int)
- [ostreambuf\\_iterator](#) & [operator++](#) ()
- [ostreambuf\\_iterator](#) & [operator=](#) ([\\_CharT](#) \_\_c)

#### Friends

- template<typename [\\_CharT2](#)> [\\_\\_gnu\\_cxx::\\_\\_enable\\_if<\\_\\_is\\_char<\\_CharT2>::\\_\\_value, ostreambuf\\_iterator<\\_CharT2>::\\_\\_type](#) [copy](#) ([istreambuf\\_iterator](#)< [\\_CharT2](#)>, [istreambuf\\_iterator](#)< [\\_CharT2](#)>, [ostreambuf\\_iterator](#)< [\\_CharT2](#)>)

## 4.659.1 Detailed Description

```
template<typename _CharT, typename _Traits> class std::ostreambuf_iterator< _CharT, _Traits >
```

Provides output iterator semantics for streambufs.

Definition at line 402 of file stl\_algobase.h.

## 4.659.2 Member Typedef Documentation

4.659.2.1 `template<typename _CharT, typename _Traits > typedef _CharT std::ostreambuf_iterator< _CharT, _Traits >::char_type`

Public typedefs.

Definition at line 223 of file ostreambuf\_iterator.h.

4.659.2.2 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::difference_type` [inherited]

Distance between iterators is represented as this type.

Definition at line 125 of file stl\_iterator\_base\_types.h.

4.659.2.3 `typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::iterator_category` [inherited]

One of the [tag types](#).

Definition at line 121 of file stl\_iterator\_base\_types.h.

4.659.2.4 `template<typename _CharT, typename _Traits > typedef basic_ostream< _CharT, _Traits > std::ostreambuf_iterator< _CharT, _Traits >::ostream_type`

Public typedefs.

Definition at line 226 of file ostreambuf\_iterator.h.

4.659.2.5 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer` [inherited]

This type represents a pointer-to-value\_type.

Definition at line 127 of file stl\_iterator\_base\_types.h.

4.659.2.6 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference` [inherited]

This type represents a reference-to-value\_type.

Definition at line 129 of file stl\_iterator\_base\_types.h.

4.659.2.7 `template<typename _CharT, typename _Traits > typedef basic_streambuf< _CharT, _Traits > std::ostreambuf_iterator< _CharT, _Traits >::streambuf_type`

Public typedefs.

Definition at line 225 of file ostreambuf\_iterator.h.

4.659.2.8 `template<typename _CharT, typename _Traits > typedef _Traits std::ostreambuf_iterator< _CharT, _Traits >::traits_type`

Public typedefs.

Definition at line 224 of file `streambuf_iterator.h`.

4.659.2.9 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 123 of file `stl_iterator_base_types.h`.

#### 4.659.3 Constructor & Destructor Documentation

4.659.3.1 `template<typename _CharT, typename _Traits > std::ostreambuf_iterator< _CharT, _Traits >::ostreambuf_iterator( ostream_type & __s ) [inline], [noexcept]`

Construct output iterator from ostream.

Definition at line 241 of file `streambuf_iterator.h`.

4.659.3.2 `template<typename _CharT, typename _Traits > std::ostreambuf_iterator< _CharT, _Traits >::ostreambuf_iterator( streambuf_type * __s ) [inline], [noexcept]`

Construct output iterator from streambuf.

Definition at line 245 of file `streambuf_iterator.h`.

#### 4.659.4 Member Function Documentation

4.659.4.1 `template<typename _CharT, typename _Traits > bool std::ostreambuf_iterator< _CharT, _Traits >::failed( ) const [inline], [noexcept]`

Return true if previous operator=() failed.

Definition at line 275 of file `streambuf_iterator.h`.

4.659.4.2 `template<typename _CharT, typename _Traits > ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits >::operator*( ) [inline]`

Return \*this.

Definition at line 260 of file `streambuf_iterator.h`.

4.659.4.3 `template<typename _CharT, typename _Traits > ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits >::operator++( int ) [inline]`

Return \*this.

Definition at line 265 of file `streambuf_iterator.h`.

4.659.4.4 `template<typename _CharT, typename _Traits > ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits >::operator++( ) [inline]`

Return \*this.

Definition at line 270 of file `streambuf_iterator.h`.

4.659.4.5 `template<typename _CharT, typename _Traits> ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits>::operator=( _CharT __c ) [inline]`

Write character to streambuf. Calls streambuf.sputc().

Definition at line 250 of file streambuf\_iterator.h.

The documentation for this class was generated from the following files:

- [stl\\_algobase.h](#)
- [streambuf\\_iterator.h](#)

## 4.660 `std::output_iterator_tag` Struct Reference

### 4.660.1 Detailed Description

Marking output iterators.

Definition at line 92 of file stl\_iterator\_base\_types.h.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 4.661 `std::owner_less< _Tp>` Struct Template Reference

### 4.661.1 Detailed Description

`template<typename _Tp> struct std::owner_less< _Tp>`

Primary template `owner_less`.

Definition at line 523 of file shared\_ptr.h.

The documentation for this struct was generated from the following file:

- [shared\\_ptr.h](#)

## 4.662 `std::owner_less< shared_ptr< _Tp>>` Struct Template Reference

Inherits `std::_Sp_owner_less< _Tp, _Tp1>`.

### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- bool **operator()** (const `_Tp` &\_\_lhs, const `_Tp` &\_\_rhs) const
- bool **operator()** (const `_Tp` &\_\_lhs, const `_Tp1` &\_\_rhs) const
- bool **operator()** (const `_Tp1` &\_\_lhs, const `_Tp` &\_\_rhs) const

#### 4.662.1 Detailed Description

```
template<typename _Tp>struct std::owner_less< shared_ptr< _Tp > >
```

Partial specialization of `owner_less` for `shared_ptr`.

Definition at line 527 of file `shared_ptr.h`.

#### 4.662.2 Member Typedef Documentation

4.662.2.1 `typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.662.2.2 `typedef bool std::binary_function< _Tp, _Tp, bool >::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.662.2.3 `typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [shared\\_ptr.h](#)

#### 4.663 `std::owner_less< weak_ptr< _Tp > >` Struct Template Reference

Inherits `std::_Sp_owner_less< _Tp, _Tp1 >`.

##### Public Types

- `typedef _Tp first_argument_type`
- `typedef bool result_type`
- `typedef _Tp second_argument_type`

##### Public Member Functions

- `bool operator() (const _Tp &__lhs, const _Tp &__rhs) const`
- `bool operator() (const _Tp &__lhs, const _Tp1 &__rhs) const`
- `bool operator() (const _Tp1 &__lhs, const _Tp &__rhs) const`

#### 4.663.1 Detailed Description

```
template<typename _Tp>struct std::owner_less< weak_ptr< _Tp > >
```

Partial specialization of `owner_less` for `weak_ptr`.

Definition at line 533 of file `shared_ptr.h`.

## 4.663.2 Member Typedef Documentation

## 4.663.2.1 typedef \_Tp std::binary\_function&lt;\_Tp, \_Tp, bool&gt;::first\_argument\_type [inherited]

first\_argument\_type is the type of the first argument

Definition at line 117 of file stl\_function.h.

## 4.663.2.2 typedef bool std::binary\_function&lt;\_Tp, \_Tp, bool&gt;::result\_type [inherited]

result\_type is the return type

Definition at line 123 of file stl\_function.h.

## 4.663.2.3 typedef \_Tp std::binary\_function&lt;\_Tp, \_Tp, bool&gt;::second\_argument\_type [inherited]

second\_argument\_type is the type of the second argument

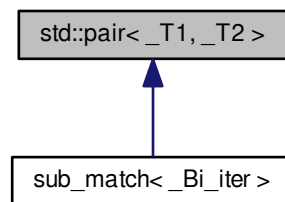
Definition at line 120 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [shared\\_ptr.h](#)

## 4.664 std::pair&lt;\_T1, \_T2&gt; Struct Template Reference

Inheritance diagram for std::pair<\_T1, \_T2>:



## Public Types

- typedef \_T1 **first\_type**
- typedef \_T2 [second\\_type](#)

## Public Member Functions

- constexpr [pair](#) ()
- constexpr [pair](#) (const \_T1 &\_\_a, const \_T2 &\_\_b)
- template<class \_U1, class \_U2, class = typename enable\_if<\_\_and<is\_convertible<const \_U1&, \_T1>, is\_convertible<const \_U2&, \_T2>>::value>::type> constexpr [pair](#) (const [pair](#)<\_U1, \_U2> &\_\_p)



- constexpr **pair** (const [pair](#) &)=default
- constexpr **pair** ([pair](#) &&)=default
- template<class \_U1 , class = typename enable\_if<is\_convertible<\_U1, \_T1>::value>::type> constexpr **pair** (\_U1 &&\_\_x, const \_T2 &\_\_y)
- template<class \_U2 , class = typename enable\_if<is\_convertible<\_U2, \_T2>::value>::type> constexpr **pair** (const \_T1 &\_\_x, \_U2 &&\_\_y)
- template<class \_U1 , class \_U2 , class = typename enable\_if<\_\_and\_<is\_convertible<\_U1, \_T1>, is\_convertible<\_U2, \_T2>>>::value>::type> constexpr **pair** (\_U1 &&\_\_x, \_U2 &&\_\_y)
- template<class \_U1 , class \_U2 , class = typename enable\_if<\_\_and\_<is\_convertible<\_U1, \_T1>, is\_convertible<\_U2, \_T2>>>::value>::type> constexpr **pair** ([pair](#)<\_U1, \_U2 > &&\_\_p)
- template<typename... \_Args1, typename... \_Args2> **pair** ([piecewise\\_construct\\_t](#), tuple<\_Args1...>, tuple<\_Args2...>)
- [pair](#) & **operator=** (const [pair](#) &\_\_p)
- [pair](#) & **operator=** ([pair](#) &&\_\_p) noexcept(\_\_and\_< is\_nothrow\_move\_assignable<\_T1 >, is\_nothrow\_move\_assignable<\_T2 >>::value)
- template<class \_U1 , class \_U2 > [pair](#) & **operator=** (const [pair](#)<\_U1, \_U2 > &\_\_p)
- template<class \_U1 , class \_U2 > [pair](#) & **operator=** ([pair](#)<\_U1, \_U2 > &&\_\_p)
- void **swap** ([pair](#) &\_\_p) noexcept(noexcept(swap([first](#), \_\_p.first))&&noexcept(swap([second](#), \_\_p.second)))

#### Public Attributes

- [\\_T1](#) [first](#)
- [\\_T2](#) [second](#)

#### 4.664.1 Detailed Description

template<class \_T1, class \_T2>struct std::pair<\_T1, \_T2 >

Struct holding two objects of arbitrary type.

#### Template Parameters

<a href="#">_T1</a>	Type of first object.
<a href="#">_T2</a>	Type of second object.

Definition at line 96 of file stl\_pair.h.

#### 4.664.2 Member Typedef Documentation

##### 4.664.2.1 template<class \_T1, class \_T2> typedef \_T2 std::pair<\_T1, \_T2 >::second\_type

[first\\_type](#) is the first bound type

Definition at line 99 of file stl\_pair.h.

#### 4.664.3 Constructor & Destructor Documentation

##### 4.664.3.1 template<class \_T1, class \_T2> constexpr std::pair<\_T1, \_T2 >::pair ( ) [inline]

[second](#) is a copy of the second object

The default constructor creates [first](#) and [second](#) using their respective default constructors.

Definition at line 108 of file stl\_pair.h.

4.664.3.2 `template<class _T1, class _T2> constexpr std::pair<_T1, _T2>::pair ( const _T1 & __a, const _T2 & __b )`  
`[inline]`

Two objects may be passed to a `pair` constructor to be copied.

Definition at line 112 of file `stl_pair.h`.

4.664.3.3 `template<class _T1, class _T2> template<class _U1, class _U2, class = typename enable_if<__and<is_↵  
 convertible<const _U1&, _T1>, is_convertible<const _U2&, _T2>>::value>::type> constexpr std::pair<_T1, _T2  
 >::pair ( const pair<_U1, _U2> & __p ) [inline]`

There is also a templated copy ctor for the `pair` class itself.

Definition at line 124 of file `stl_pair.h`.

#### 4.664.4 Member Data Documentation

4.664.4.1 `template<class _T1, class _T2> _T1 std::pair<_T1, _T2>::first`

`second_type` is the second bound type

Definition at line 101 of file `stl_pair.h`.

Referenced by `std::_Temporary_buffer<_ForwardIterator, _Tp>::_Temporary_buffer()`, `std::set<_StateIdT>::insert()`, `std::operator<()`, and `std::operator==()`.

4.664.4.2 `template<class _T1, class _T2> _T2 std::pair<_T1, _T2>::second`

`first` is a copy of the first object

Definition at line 102 of file `stl_pair.h`.

Referenced by `std::_Temporary_buffer<_ForwardIterator, _Tp>::_Temporary_buffer()`, `std::set<_StateIdT>::insert()`, and `std::operator==()`.

The documentation for this struct was generated from the following file:

- [stl\\_pair.h](#)

## 4.665 `std::piecewise_constant_distribution<_RealType>` Class Template Reference

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

### Public Member Functions

- `template<typename _InputIteratorB, typename _InputIteratorW> piecewise_constant_distribution ( _InputIteratorB __↵  
 bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin)`
- `template<typename _Func> piecewise_constant_distribution (initializer_list<_RealType> __bl, _Func __fw)`
- `template<typename _Func> piecewise_constant_distribution (size_t __nw, _RealType __xmin, _RealType __↵  
 xmax, _Func __fw)`

- **piecewise\_constant\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename [\\_ForwardIterator](#) , typename [\\_UniformRandomNumberGenerator](#) > void **\_\_generate** ([\\_ForwardIterator](#) \_\_f, [\\_ForwardIterator](#) \_\_t, [\\_UniformRandomNumberGenerator](#) &\_\_urng)
- template<typename [\\_ForwardIterator](#) , typename [\\_UniformRandomNumberGenerator](#) > void **\_\_generate** ([\\_ForwardIterator](#) \_\_f, [\\_ForwardIterator](#) \_\_t, [\\_UniformRandomNumberGenerator](#) &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename [\\_UniformRandomNumberGenerator](#) > void **\_\_generate** ([result\\_type](#) \* \_\_f, [result\\_type](#) \* \_\_t, [\\_UniformRandomNumberGenerator](#) &\_\_urng, const [param\\_type](#) &\_\_p)
- [std::vector](#)< double > **densities** () const
- [std::vector](#)< [\\_RealType](#) > **intervals** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename [\\_UniformRandomNumberGenerator](#) > [result\\_type](#) **operator()** ([\\_UniformRandomNumberGenerator](#) &\_\_urng)
- template<typename [\\_UniformRandomNumberGenerator](#) > [result\\_type](#) **operator()** ([\\_UniformRandomNumberGenerator](#) &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

#### Friends

- template<typename [\\_RealType1](#) , typename [\\_CharT](#) , typename [\\_Traits](#) > [std::basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > & **operator<<** ([std::basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > &\_\_os, const [std::piecewise\\_constant\\_distribution](#)< [\\_RealType1](#) > &\_\_x)
- bool **operator==** (const [piecewise\\_constant\\_distribution](#) &\_\_d1, const [piecewise\\_constant\\_distribution](#) &\_\_d2)
- template<typename [\\_RealType1](#) , typename [\\_CharT](#) , typename [\\_Traits](#) > [std::basic\\_istream](#)< [\\_CharT](#), [\\_Traits](#) > & **operator>>** ([std::basic\\_istream](#)< [\\_CharT](#), [\\_Traits](#) > &\_\_is, [std::piecewise\\_constant\\_distribution](#)< [\\_RealType1](#) > &\_\_x)

#### 4.665.1 Detailed Description

```
template<typename \_RealType = double>class std::piecewise\_constant\_distribution< \_RealType >
```

A [piecewise\\_constant\\_distribution](#) random number distribution.

The formula for the piecewise constant probability mass function is

Definition at line 5481 of file [random.h](#).

#### 4.665.2 Member Typedef Documentation

4.665.2.1 `template<typename \_RealType = double> typedef \_RealType std::piecewise\_constant\_distribution< \_RealType >::result\_type`

The type of the range of the distribution.

Definition at line 5484 of file [random.h](#).

#### 4.665.3 Member Function Documentation

4.665.3.1 `template<typename \_RealType = double> std::vector<double> std::piecewise\_constant\_distribution< \_RealType >::densities ( ) const` `[inline]`

Returns a vector of the probability densities.

Definition at line 5602 of file random.h.

References std::vector<\_Tp, \_Alloc>::empty().

**4.665.3.2** `template<typename _RealType = double> std::vector<_RealType> std::piecewise_constant_distribution<_RealType>::intervals ( ) const [inline]`

Returns a vector of the intervals.

Definition at line 5586 of file random.h.

References std::vector<\_Tp, \_Alloc>::empty().

**4.665.3.3** `template<typename _RealType = double> result_type std::piecewise_constant_distribution<_RealType>::max ( ) const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 5637 of file random.h.

References std::vector<\_Tp, \_Alloc>::back(), and std::vector<\_Tp, \_Alloc>::empty().

**4.665.3.4** `template<typename _RealType = double> result_type std::piecewise_constant_distribution<_RealType>::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 5627 of file random.h.

References std::vector<\_Tp, \_Alloc>::empty(), and std::vector<\_Tp, \_Alloc>::front().

**4.665.3.5** `template<typename _RealType = double> template<typename UniformRandomNumberGenerator> result_type std::piecewise_constant_distribution<_RealType>::operator() ( UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 5648 of file random.h.

**4.665.3.6** `template<typename _RealType = double> param_type std::piecewise_constant_distribution<_RealType>::param ( ) const [inline]`

Returns the parameter set of the distribution.

Definition at line 5612 of file random.h.

**4.665.3.7** `template<typename _RealType = double> void std::piecewise_constant_distribution<_RealType>::param ( const param_type & __param ) [inline]`

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 5620 of file random.h.

**4.665.3.8** `template<typename _RealType = double> void std::piecewise_constant_distribution<_RealType>::reset ( ) [inline]`

Resets the distribution state.

Definition at line 5579 of file random.h.

#### 4.665.4 Friends And Related Function Documentation

4.665.4.1 `template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits  
> std::basic_ostream<_CharT, _Traits>& operator<< ( std::basic_ostream<_CharT, _Traits> & __os, const  
std::piecewise_constant_distribution<_RealType1> & __x ) [friend]`

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

##### Parameters

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

##### Returns

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

4.665.4.2 `template<typename _RealType = double> bool operator==( const piecewise_constant_distribution<_RealType  
> & __d1, const piecewise_constant_distribution<_RealType> & __d2 ) [friend]`

Return true if two piecewise constant distributions have the same parameters.

Definition at line 5683 of file random.h.

4.665.4.3 `template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename  
_Traits> std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<_CharT, _Traits> & __is,  
std::piecewise_constant_distribution<_RealType1> & __x ) [friend]`

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

##### Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>piecewise_constant_distribution</code> random number generator engine.

##### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

#### 4.666 std::piecewise\_constant\_distribution<\_RealType>::param\_type Struct Reference

##### Public Types

- typedef `piecewise_constant_distribution<_RealType>` **distribution\_type**

##### Public Member Functions

- `template<typename _InputIteratorB , typename _InputIteratorW> param_type ( _InputIteratorB __bfirst, _InputIteratorB __  
__bend, _InputIteratorW __wbegin)`

- template<typename \_Func > **param\_type** (initializer\_list< \_RealType > \_\_bi, \_Func \_\_fw)
- template<typename \_Func > **param\_type** (size\_t \_\_nw, \_RealType \_\_xmin, \_RealType \_\_xmax, \_Func \_\_fw)
- **param\_type** (const [param\\_type](#) &)=default
- [std::vector](#)< double > **densities** () const
- [std::vector](#)< \_RealType > **intervals** () const
- [param\\_type](#) & **operator=** (const [param\\_type](#) &)=default

#### Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- class **piecewise\_constant\_distribution**< \_RealType >

#### 4.666.1 Detailed Description

template<typename \_RealType = double>struct std::piecewise\_constant\_distribution< \_RealType >::param\_type

Parameter type.

Definition at line 5490 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 4.667 std::piecewise\_construct\_t Struct Reference

#### 4.667.1 Detailed Description

piecewise\_construct\_t

Definition at line 76 of file stl\_pair.h.

The documentation for this struct was generated from the following file:

- [stl\\_pair.h](#)

## 4.668 std::piecewise\_linear\_distribution< \_RealType > Class Template Reference

#### Classes

- struct [param\\_type](#)

#### Public Types

- typedef \_RealType [result\\_type](#)

#### Public Member Functions

- template<typename \_InputIteratorB , typename \_InputIteratorW > **piecewise\_linear\_distribution** (\_InputIteratorB \_\_bfirst, \_InputIteratorB \_\_bend, \_InputIteratorW \_\_wbegin)
- template<typename \_Func > **piecewise\_linear\_distribution** (initializer\_list< \_RealType > \_\_bl, \_Func \_\_fw)

- `template<typename _Func > piecewise_linear_distribution (size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw)`
- `piecewise_linear_distribution (const param\_type &__p)`
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator > void __generate (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)`
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator > void __generate (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param\_type &__p)`
- `template<typename _UniformRandomNumberGenerator > void __generate (result\_type * __f, result\_type * __t, _UniformRandomNumberGenerator &__urng, const param\_type &__p)`
- `std::vector< double > densities () const`
- `std::vector< _RealType > intervals () const`
- `result\_type max () const`
- `result\_type min () const`
- `template<typename _UniformRandomNumberGenerator > result\_type operator() (_UniformRandomNumberGenerator &__urng)`
- `template<typename _UniformRandomNumberGenerator > result\_type operator() (_UniformRandomNumberGenerator &__urng, const param\_type &__p)`
- `param\_type param () const`
- `void param (const param\_type &__param)`
- `void reset ()`

#### Friends

- `template<typename _RealType1, typename _CharT, typename _Traits > std::basic\_ostream< _CharT, _Traits > & operator<< (std::basic\_ostream< _CharT, _Traits > &__os, const std::piecewise\_linear\_distribution< _RealType1 > &__x)`
- `bool operator== (const piecewise\_linear\_distribution &__d1, const piecewise\_linear\_distribution &__d2)`
- `template<typename _RealType1, typename _CharT, typename _Traits > std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _CharT, _Traits > &__is, std::piecewise\_linear\_distribution< _RealType1 > &__x)`

#### 4.668.1 Detailed Description

```
template<typename _RealType = double>class std::piecewise_linear_distribution< _RealType >
```

A `piecewise_linear_distribution` random number distribution.

The formula for the piecewise linear probability mass function is

Definition at line 5748 of file `random.h`.

#### 4.668.2 Member Typedef Documentation

4.668.2.1 `template<typename _RealType = double> typedef _RealType std::piecewise\_linear\_distribution< _RealType >::result\_type`

The type of the range of the distribution.

Definition at line 5751 of file `random.h`.

## 4.668.3 Member Function Documentation

4.668.3.1 `template<typename _RealType = double> std::vector<double> std::piecewise_linear_distribution<_RealType>::densities ( ) const [inline]`

Return a vector of the probability densities of the distribution.

Definition at line 5872 of file random.h.

References `std::vector<_Tp, _Alloc>::empty()`.

4.668.3.2 `template<typename _RealType = double> std::vector<_RealType> std::piecewise_linear_distribution<_RealType>::intervals ( ) const [inline]`

Return the intervals of the distribution.

Definition at line 5855 of file random.h.

References `std::vector<_Tp, _Alloc>::empty()`.

4.668.3.3 `template<typename _RealType = double> result_type std::piecewise_linear_distribution<_RealType>::max ( ) const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 5907 of file random.h.

References `std::vector<_Tp, _Alloc>::back()`, and `std::vector<_Tp, _Alloc>::empty()`.

4.668.3.4 `template<typename _RealType = double> result_type std::piecewise_linear_distribution<_RealType>::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 5897 of file random.h.

References `std::vector<_Tp, _Alloc>::empty()`, and `std::vector<_Tp, _Alloc>::front()`.

4.668.3.5 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator> result_type std::piecewise_linear_distribution<_RealType>::operator() ( _UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 5918 of file random.h.

4.668.3.6 `template<typename _RealType = double> param_type std::piecewise_linear_distribution<_RealType>::param ( ) const [inline]`

Returns the parameter set of the distribution.

Definition at line 5882 of file random.h.

4.668.3.7 `template<typename _RealType = double> void std::piecewise_linear_distribution<_RealType>::param ( const param_type & __param ) [inline]`

Sets the parameter set of the distribution.



## Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 5890 of file random.h.

4.668.3.8 `template<typename _RealType = double> void std::piecewise_linear_distribution<_RealType>::reset ( )`  
`[inline]`

Resets the distribution state.

Definition at line 5848 of file random.h.

## 4.668.4 Friends And Related Function Documentation

4.668.4.1 `template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits`  
`> std::basic_ostream<_CharT, _Traits>& operator<< ( std::basic_ostream<_CharT, _Traits> & __os, const`  
`std::piecewise_linear_distribution<_RealType1> & __x ) [friend]`

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

## Parameters

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

## Returns

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

4.668.4.2 `template<typename _RealType = double> bool operator==( const piecewise_linear_distribution<_RealType> &`  
`__d1, const piecewise_linear_distribution<_RealType> & __d2 ) [friend]`

Return true if two `piecewise_linear_distribution` have the same parameters.

Definition at line 5953 of file random.h.

4.668.4.3 `template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename`  
`_Traits > std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<_CharT, _Traits> & __is,`  
`std::piecewise_linear_distribution<_RealType1> & __x ) [friend]`

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

## Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>piecewise_linear_distribution</code> random number generator engine.

## Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

4.669 `std::piecewise_linear_distribution<_RealType>::param_type` Struct Reference

## Public Types

- typedef `piecewise_linear_distribution<_RealType>` **distribution\_type**

## Public Member Functions

- template<typename \_InputIteratorB, typename \_InputIteratorW> **param\_type** (\_InputIteratorB \_\_bfirst, \_InputIteratorB \_\_bend, \_InputIteratorW \_\_wbegin)
- template<typename \_Func> **param\_type** (initializer\_list<\_RealType> \_\_bl, \_Func \_\_fw)
- template<typename \_Func> **param\_type** (size\_t \_\_nw, \_RealType \_\_xmin, \_RealType \_\_xmax, \_Func \_\_fw)
- **param\_type** (const `param_type` &)=default
- `std::vector<double>` **densities** () const
- `std::vector<_RealType>` **intervals** () const
- `param_type` & **operator=** (const `param_type` &)=default

## Friends

- bool **operator==** (const `param_type` &\_\_p1, const `param_type` &\_\_p2)
- class `piecewise_linear_distribution<_RealType>`

## 4.669.1 Detailed Description

template<typename \_RealType = double>struct std::piecewise\_linear\_distribution<\_RealType>::param\_type

Parameter type.

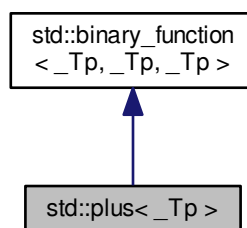
Definition at line 5757 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.670 `std::plus<_Tp>` Struct Template Reference

Inheritance diagram for `std::plus<_Tp>`:



## Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Tp` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

## Public Member Functions

- `_Tp` **operator()** (const `_Tp` &\_\_x, const `_Tp` &\_\_y) const

### 4.670.1 Detailed Description

template<typename `_Tp`>struct std::plus< `_Tp` >

One of the [math functors](#).

Definition at line 140 of file `stl_function.h`.

### 4.670.2 Member Typedef Documentation

4.670.2.1 typedef `_Tp` std::binary\_function< `_Tp`, `_Tp`, `_Tp` >::first\_argument\_type [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.670.2.2 typedef `_Tp` std::binary\_function< `_Tp`, `_Tp`, `_Tp` >::result\_type [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.670.2.3 typedef `_Tp` std::binary\_function< `_Tp`, `_Tp`, `_Tp` >::second\_argument\_type [inherited]

`second_argument_type` is the type of the second argument

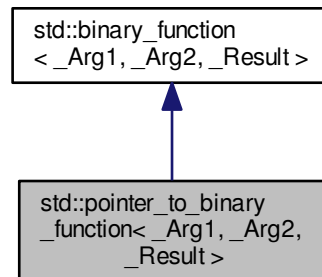
Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.671 std::pointer\_to\_binary\_function&lt; \_Arg1, \_Arg2, \_Result &gt; Class Template Reference

Inheritance diagram for std::pointer\_to\_binary\_function< \_Arg1, \_Arg2, \_Result >:



## Public Types

- typedef \_Arg1 [first\\_argument\\_type](#)
- typedef \_Result [result\\_type](#)
- typedef \_Arg2 [second\\_argument\\_type](#)

## Public Member Functions

- **pointer\_to\_binary\_function** (\_Result(\*\_\_x)(\_Arg1, \_Arg2))
- \_Result **operator()** (\_Arg1 \_\_x, \_Arg2 \_\_y) const

## Protected Attributes

- \_Result(\* **M\_ptr**)(\_Arg1, \_Arg2)

## 4.671.1 Detailed Description

```
template<typename _Arg1, typename _Arg2, typename _Result>class std::pointer_to_binary_function< _Arg1, _Arg2, _Result >
```

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

Definition at line 447 of file stl\_function.h.

## 4.671.2 Member Typedef Documentation

4.671.2.1 `template<typename _Arg1, typename _Arg2, typename _Result> typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result >::first_argument_type` `[inherited]`

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl\_function.h.

4.671.2.2 `template<typename _Arg1, typename _Arg2, typename _Result> typedef _Result std::binary_function< _Arg1, _Arg2, _Result >::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.671.2.3 `template<typename _Arg1, typename _Arg2, typename _Result> typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result >::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

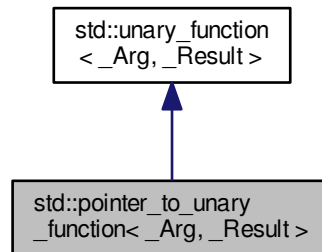
Definition at line 120 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 4.672 `std::pointer_to_unary_function< _Arg, _Result >` Class Template Reference

Inheritance diagram for `std::pointer_to_unary_function< _Arg, _Result >`:



### Public Types

- `typedef _Arg` [argument\\_type](#)
- `typedef _Result` [result\\_type](#)

### Public Member Functions

- `pointer_to_unary_function` (`_Result`(\*\_\_x)(\_\_Arg))
- `_Result operator()` (`_Arg __x`) const

### Protected Attributes

- `_Result`(\* `_M_ptr`)(\_\_Arg)

## 4.672.1 Detailed Description

```
template<typename _Arg, typename _Result> class std::pointer_to_unary_function< _Arg, _Result >
```

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

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

## 4.672.2 Member Typedef Documentation

```
4.672.2.1 template<typename _Arg, typename _Result> typedef _Arg std::unary_function< _Arg, _Result
>::argument_type [inherited]
```

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

```
4.672.2.2 template<typename _Arg, typename _Result> typedef _Result std::unary_function< _Arg, _Result >::result_type
[inherited]
```

`result_type` is the return type

Definition at line 107 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

4.673 `std::pointer_traits<_Ptr>` Struct Template Reference

Inherits `std::__ptrr_pointer_to<_Ptr>`.

## Public Types

- typedef `__ptrr_diff_type<_Ptr>::__type` [difference\\_type](#)
- typedef `__ptrr_elt_type<_Ptr>::__type` [element\\_type](#)
- typedef `_Ptr` [pointer](#)
- template<typename `_Up`> using **rebind** = `typename __ptrr_rebind<_Ptr, _Up>::__type`

## Static Public Member Functions

- static `_Ptr` **pointer\_to** (`__element_type &__e`)

## 4.673.1 Detailed Description

```
template<typename _Ptr> struct std::pointer_traits<_Ptr>
```

Uniform interface to all pointer-like types.

Definition at line 137 of file `ptr_traits.h`.

#### 4.673.2 Member Typedef Documentation

4.673.2.1 `template<typename _Ptr> typedef __ptrtr_diff_type<_Ptr>::__type std::pointer_traits<_Ptr>::difference_type`

Type used to represent the difference between two pointers.

Definition at line 144 of file `ptr_traits.h`.

4.673.2.2 `template<typename _Ptr> typedef __ptrtr_elt_type<_Ptr>::__type std::pointer_traits<_Ptr>::element_type`

The type pointed to.

Definition at line 142 of file `ptr_traits.h`.

4.673.2.3 `template<typename _Ptr> typedef _Ptr std::pointer_traits<_Ptr>::pointer`

The pointer type.

Definition at line 140 of file `ptr_traits.h`.

The documentation for this struct was generated from the following file:

- [ptr\\_traits.h](#)

#### 4.674 `std::pointer_traits<_Tp*>` Struct Template Reference

##### Public Types

- typedef `ptrdiff_t` [difference\\_type](#)
- typedef `_Tp` [element\\_type](#)
- typedef `_Tp*` [pointer](#)
- `template<typename _Up> using rebind = _Up*`

##### Static Public Member Functions

- static [pointer](#) [pointer\\_to](#) (`typename __ptrtr_not_void< element\_type>::__type &__r`) noexcept

#### 4.674.1 Detailed Description

`template<typename _Tp> struct std::pointer_traits<_Tp*>`

Partial specialization for built-in pointers.

Definition at line 155 of file `ptr_traits.h`.

#### 4.674.2 Member Typedef Documentation

4.674.2.1 `template<typename _Tp> typedef ptrdiff_t std::pointer_traits<_Tp*>::difference_type`

Type used to represent the difference between two pointers.

Definition at line 162 of file `ptr_traits.h`.

4.674.2.2 `template<typename _Tp> typedef _Tp std::pointer_traits<_Tp*>::element_type`

The type pointed to.

Definition at line 160 of file `ptr_traits.h`.

4.674.2.3 `template<typename _Tp> typedef _Tp* std::pointer_traits<_Tp*>::pointer`

The pointer type.

Definition at line 158 of file `ptr_traits.h`.

#### 4.674.3 Member Function Documentation

4.674.3.1 `template<typename _Tp> static pointer std::pointer_traits<_Tp*>::pointer_to ( typename __ptrtr_not_void<element_type>::__type &__r ) [inline],[static],[noexcept]`

Obtain a pointer to an object.

##### Parameters

<code>__r</code>	A reference to an object of type <code>element_type</code>
------------------	--

##### Returns

`addressof (__r)`

Definition at line 173 of file `ptr_traits.h`.

References `std::addressof()`.

The documentation for this struct was generated from the following file:

- [ptr\\_traits.h](#)

## 4.675 `std::poisson_distribution<_IntType>` Class Template Reference

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_IntType` [result\\_type](#)

### Public Member Functions

- **`poisson_distribution`** (`double __mean=1.0`)
- **`poisson_distribution`** (`const param\_type &__p`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator> void __generate (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)`
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator> void __generate (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param\_type &__p)`
- `template<typename _UniformRandomNumberGenerator> void __generate (result\_type *__f, result\_type *__t, _UniformRandomNumberGenerator &__urng, const param\_type &__p)`



- `result_type max () const`
- `double mean () const`
- `result_type min () const`
- `template<typename _UniformRandomNumberGenerator > result_type operator() (_UniformRandomNumberGenerator &&__urng)`
- `template<typename _UniformRandomNumberGenerator > result_type operator() (_UniformRandomNumberGenerator &&__urng, const param_type &__p)`
- `param_type param () const`
- `void param (const param_type &__param)`
- `void reset ()`

#### Friends

- `template<typename _IntType1, typename _CharT, typename _Traits > std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::poisson_distribution< _IntType1 > &__x)`
- `bool operator== (const poisson_distribution &__d1, const poisson_distribution &__d2)`
- `template<typename _IntType1, typename _CharT, typename _Traits > std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::poisson_distribution< _IntType1 > &__x)`

#### 4.675.1 Detailed Description

`template<typename _IntType = int> class std::poisson_distribution< _IntType >`

A discrete Poisson random number distribution.

The formula for the Poisson probability density function is  $p(i|\mu) = \frac{\mu^i}{i!} e^{-\mu}$  where  $\mu$  is the parameter of the distribution.

Definition at line 4430 of file random.h.

#### 4.675.2 Member Typedef Documentation

4.675.2.1 `template<typename _IntType = int> typedef _IntType std::poisson_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 4433 of file random.h.

#### 4.675.3 Member Function Documentation

4.675.3.1 `template<typename _IntType = int> result_type std::poisson_distribution< _IntType >::max ( ) const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 4524 of file random.h.

References `std::max()`.

4.675.3.2 `template<typename _IntType = int> double std::poisson_distribution< _IntType >::mean ( ) const [inline]`

Returns the distribution parameter `mean`.

Definition at line 4495 of file random.h.

4.675.3.3 `template<typename _IntType = int> result_type std::poisson_distribution< _IntType >::min ( ) const`  
`[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4517 of file random.h.

4.675.3.4 `template<typename _IntType = int> template<typename _UniformRandomNumberGenerator > result_type`  
`std::poisson_distribution< _IntType >::operator() ( _UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 4532 of file random.h.

4.675.3.5 `template<typename _IntType = int> param_type std::poisson_distribution< _IntType >::param ( ) const`  
`[inline]`

Returns the parameter set of the distribution.

Definition at line 4502 of file random.h.

4.675.3.6 `template<typename _IntType = int> void std::poisson_distribution< _IntType >::param ( const param_type &`  
`__param ) [inline]`

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 4510 of file random.h.

4.675.3.7 `template<typename _IntType = int> void std::poisson_distribution< _IntType >::reset ( ) [inline]`

Resets the distribution state.

Definition at line 4488 of file random.h.

References `std::normal_distribution< _RealType >::reset()`.

#### 4.675.4 Friends And Related Function Documentation

4.675.4.1 `template<typename _IntType = int> template<typename _IntType1 , typename _CharT , typename _Traits >`  
`std::basic_ostream< _CharT, _Traits>& operator<< ( std::basic_ostream< _CharT, _Traits > & __os, const`  
`std::poisson_distribution< _IntType1 > & __x ) [friend]`

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

Parameters

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

**Returns**

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

4.675.4.2 `template<typename _IntType = int> bool operator==( const poisson_distribution< _IntType > &__d1, const poisson_distribution< _IntType > &__d2 ) [friend]`

Return true if two Poisson distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 4568 of file random.h.

4.675.4.3 `template<typename _IntType = int> template<typename _IntType1, typename _CharT, typename _Traits > std::basic_istream< _CharT, _Traits>& operator>> ( std::basic_istream< _CharT, _Traits > &__is, std::poisson_distribution< _IntType1 > &__x ) [friend]`

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

**Parameters**

<code>__is</code>	An input stream.
<code>__x</code>	A <code>poisson_distribution</code> random number generator engine.

**Returns**

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

**4.676 std::poisson\_distribution< \_IntType >::param\_type Struct Reference****Public Types**

- typedef `poisson_distribution< _IntType >` **distribution\_type**

**Public Member Functions**

- **param\_type** (double \_\_mean=1.0)
- double **mean** () const

**Friends**

- bool **operator==(** (const `param_type` &\_\_p1, const `param_type` &\_\_p2)
- class **poisson\_distribution< \_IntType >**

**4.676.1 Detailed Description**

`template<typename _IntType = int> struct std::poisson_distribution< _IntType >::param_type`

Parameter type.

Definition at line 4439 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 4.677 `std::priority_queue< _Tp, _Sequence, _Compare >` Class Template Reference

### Public Types

- typedef `_Sequence::const_reference` **const\_reference**
- typedef `_Sequence` **container\_type**
- typedef `_Sequence::reference` **reference**
- typedef `_Sequence::size_type` **size\_type**
- typedef `_Sequence::value_type` **value\_type**

### Public Member Functions

- [priority\\_queue](#) (const `_Compare` &\_\_x, const `_Sequence` &\_\_s)
- **priority\_queue** (const `_Compare` &\_\_x=`_Compare`(), `_Sequence` &&\_\_s=`_Sequence`())
- template<typename `_InputIterator` > [priority\\_queue](#) (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_x, const `_Sequence` &\_\_s)
- template<typename `_InputIterator` > **priority\_queue** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_x=`_Compare`(), `_Sequence` &&\_\_s=`_Sequence`())
- template<typename... `_Args`> void **emplace** (`_Args` &&...\_\_args)
- bool [empty](#) () const
- void [pop](#) ()
- void [push](#) (const `value_type` &\_\_x)
- void **push** (`value_type` &&\_\_x)
- `size_type` [size](#) () const
- void **swap** ([priority\\_queue](#) &\_\_pq) noexcept(noexcept(swap(c, \_\_pq.c))&&noexcept(swap(comp, \_\_pq.comp)))
- const\_reference [top](#) () const

### Protected Attributes

- `_Sequence` **c**
- `_Compare` **comp**

### 4.677.1 Detailed Description

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>>
class std::priority_queue< _Tp, _Sequence, _Compare >
```

A standard container automatically sorting its contents.

### Template Parameters

<code>_Tp</code>	Type of element.
<code>_Sequence</code>	Type of underlying sequence, defaults to <code>vector&lt;_Tp&gt;</code> .
<code>_Compare</code>	Comparison function object type, defaults to <code>less&lt;_Sequence::value_type&gt;</code> .

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces priority-based sorting and queue behavior. Very few of the standard container/sequence interface requirements are met (e.g., iterators).

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::vector`, but it can be any type that supports `front()`, `push_back`, `pop_back`, and random-access iterators, such as `std::deque` or an appropriate user-defined type.

The third template parameter supplies the means of making priority comparisons. It defaults to `less<value_type>` but can be anything defining a strict weak ordering.

Members not found in *normal* containers are `container_type`, which is a typedef for the second `Sequence` parameter, and `push`, `pop`, and `top`, which are standard queue operations.

#### Note

No equality/comparison operators are provided for `priority_queue`.

Sorting of the elements takes place as they are added to, and removed from, the `priority_queue` using the `priority_queue`'s member functions. If you access the elements by other means, and change their data such that the sorting order would be different, the `priority_queue` will not re-sort the elements for you. (How could it know to do so?)

Definition at line 367 of file `stl_queue.h`.

### 4.677.2 Constructor & Destructor Documentation

4.677.2.1 `template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>> std::priority_queue<_Tp, _Sequence, _Compare>::priority_queue ( const _Compare &__x, const _Sequence &__s ) [inline], [explicit]`

Default constructor creates no elements.

Definition at line 402 of file `stl_queue.h`.

References `std::make_heap()`.

4.677.2.2 `template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>> template<typename _InputIterator > std::priority_queue<_Tp, _Sequence, _Compare>::priority_queue ( _InputIterator __first, _InputIterator __last, const _Compare &__x, const _Sequence &__s ) [inline]`

Builds a queue from a range.

#### Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__x</code>	A comparison functor describing a strict weak ordering.
<code>__s</code>	An initial sequence with which to start.

Begins by copying `__s`, inserting a copy of the elements from `[first,last)` into the copy of `__s`, then ordering the copy according to `__x`.

For more information on function objects, see the documentation on [functor base classes](#).

Definition at line 442 of file `stl_queue.h`.

References std::make\_heap().

#### 4.677.3 Member Function Documentation

**4.677.3.1** `template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename  
_Sequence::value_type>> bool std::priority_queue<_Tp, _Sequence, _Compare >::empty ( ) const  
[inline]`

Returns true if the queue is empty.

Definition at line 468 of file stl\_queue.h.

Referenced by \_\_gnu\_parallel::multiseq\_partition(), and \_\_gnu\_parallel::multiseq\_selection().

**4.677.3.2** `template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename  
_Sequence::value_type>> void std::priority_queue<_Tp, _Sequence, _Compare >::pop ( ) [inline]`

Removes first element.

This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop() is called.

Definition at line 531 of file stl\_queue.h.

References std::pop\_heap().

Referenced by \_\_gnu\_parallel::multiseq\_partition(), and \_\_gnu\_parallel::multiseq\_selection().

**4.677.3.3** `template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename  
_Sequence::value_type>> void std::priority_queue<_Tp, _Sequence, _Compare >::push ( const value_type & __x  
) [inline]`

Add data to the queue.

##### Parameters

<code>__x</code>	Data to be added.
------------------	-------------------

This is a typical queue operation. The time complexity of the operation depends on the underlying sequence.

Definition at line 496 of file stl\_queue.h.

References std::push\_heap().

Referenced by \_\_gnu\_parallel::multiseq\_partition(), and \_\_gnu\_parallel::multiseq\_selection().

**4.677.3.4** `template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename  
_Sequence::value_type>> size_type std::priority_queue<_Tp, _Sequence, _Compare >::size ( ) const  
[inline]`

Returns the number of elements in the queue.

Definition at line 473 of file stl\_queue.h.

**4.677.3.5** `template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename  
_Sequence::value_type>> const_reference std::priority_queue<_Tp, _Sequence, _Compare >::top ( ) const  
[inline]`

Returns a read-only (constant) reference to the data at the first element of the queue.

Definition at line 481 of file `stl_queue.h`.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

The documentation for this class was generated from the following file:

- [stl\\_queue.h](#)

## 4.678 `std::queue<_Tp, _Sequence>` Class Template Reference

### Public Types

- `typedef _Sequence::const_reference` **const\_reference**
- `typedef _Sequence` **container\_type**
- `typedef _Sequence::reference` **reference**
- `typedef _Sequence::size_type` **size\_type**
- `typedef _Sequence::value_type` **value\_type**

### Public Member Functions

- [queue](#) (const \_Sequence &\_\_c)
- **queue** (\_Sequence &&\_\_c=\_Sequence())
- reference [back](#) ()
- const\_reference [back](#) () const
- `template<typename... _Args> void` **emplace** (\_Args &&... \_\_args)
- bool [empty](#) () const
- reference [front](#) ()
- const\_reference [front](#) () const
- void [pop](#) ()
- void [push](#) (const value\_type &\_\_x)
- void **push** (value\_type &&\_\_x)
- size\_type [size](#) () const
- void **swap** ([queue](#) &\_\_q) noexcept(noexcept(swap(c, \_\_q.c)))

### Protected Attributes

- \_Sequence [c](#)

### Friends

- `template<typename _Tp1, typename _Seq1> bool` **operator<** (const [queue](#)<\_Tp1, \_Seq1> &, const [queue](#)<\_Tp1, \_Seq1> &)
- `template<typename _Tp1, typename _Seq1> bool` **operator==** (const [queue](#)<\_Tp1, \_Seq1> &, const [queue](#)<\_Tp1, \_Seq1> &)

#### 4.678.1 Detailed Description

`template<typename _Tp, typename _Sequence = deque<_Tp>> class std::queue<_Tp, _Sequence>`

A standard container giving FIFO behavior.

## Template Parameters

<code>_Tp</code>	Type of element.
<code>_Sequence</code>	Type of underlying sequence, defaults to <code>deque&lt;_Tp&gt;</code> .

Meets many of the requirements of a `container`, but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-first-out queue behavior.

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::deque`, but it can be any type that supports `front`, `back`, `push_back`, and `pop_front`, such as `std::list` or an appropriate user-defined type.

Members not found in *normal* containers are `container_type`, which is a typedef for the second `Sequence` parameter, and `push` and `pop`, which are standard queue/FIFO operations.

Definition at line 93 of file `stl_queue.h`.

## 4.678.2 Constructor &amp; Destructor Documentation

4.678.2.1 `template<typename _Tp, typename _Sequence = deque<_Tp>> std::queue<_Tp, _Sequence>::queue ( const _Sequence &__c ) [inline], [explicit]`

Default constructor creates no elements.

Definition at line 138 of file `stl_queue.h`.

## 4.678.3 Member Function Documentation

4.678.3.1 `template<typename _Tp, typename _Sequence = deque<_Tp>> reference std::queue<_Tp, _Sequence>::back ( ) [inline]`

Returns a read/write reference to the data at the last element of the queue.

Definition at line 185 of file `stl_queue.h`.

4.678.3.2 `template<typename _Tp, typename _Sequence = deque<_Tp>> const_reference std::queue<_Tp, _Sequence>::back ( ) const [inline]`

Returns a read-only (constant) reference to the data at the last element of the queue.

Definition at line 196 of file `stl_queue.h`.

4.678.3.3 `template<typename _Tp, typename _Sequence = deque<_Tp>> bool std::queue<_Tp, _Sequence>::empty ( ) const [inline]`

Returns true if the queue is empty.

Definition at line 150 of file `stl_queue.h`.

4.678.3.4 `template<typename _Tp, typename _Sequence = deque<_Tp>> reference std::queue<_Tp, _Sequence>::front ( ) [inline]`

Returns a read/write reference to the data at the first element of the queue.

Definition at line 163 of file `stl_queue.h`.



**4.678.3.5** `template<typename _Tp, typename _Sequence = deque<_Tp>> const_reference std::queue<_Tp, _Sequence>::front ( ) const [inline]`

Returns a read-only (constant) reference to the data at the first element of the queue.

Definition at line 174 of file `stl_queue.h`.

**4.678.3.6** `template<typename _Tp, typename _Sequence = deque<_Tp>> void std::queue<_Tp, _Sequence>::pop ( ) [inline]`

Removes first element.

This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop()` is called.

Definition at line 238 of file `stl_queue.h`.

**4.678.3.7** `template<typename _Tp, typename _Sequence = deque<_Tp>> void std::queue<_Tp, _Sequence>::push ( const value_type &__x ) [inline]`

Add data to the end of the queue.

Parameters

<code>__x</code>	Data to be added.
------------------	-------------------

This is a typical queue operation. The function creates an element at the end of the queue and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.

Definition at line 212 of file `stl_queue.h`.

**4.678.3.8** `template<typename _Tp, typename _Sequence = deque<_Tp>> size_type std::queue<_Tp, _Sequence>::size ( ) const [inline]`

Returns the number of elements in the queue.

Definition at line 155 of file `stl_queue.h`.

#### 4.678.4 Member Data Documentation

**4.678.4.1** `template<typename _Tp, typename _Sequence = deque<_Tp>> _Sequence std::queue<_Tp, _Sequence>::c [protected]`

'c' is the underlying container. Maintainers wondering why this isn't uglified as per style guidelines should note that this name is specified in the standard, [23.2.3.1]. (Why? Presumably for the same reason that it's protected instead of private: to allow derivation. But none of the other containers allow for derivation. Odd.)

Definition at line 126 of file `stl_queue.h`.

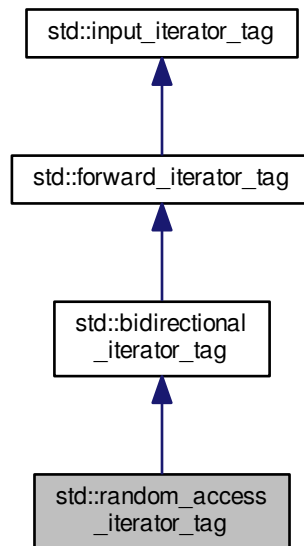
Referenced by `std::operator<()`, and `std::operator==()`.

The documentation for this class was generated from the following file:

- [stl\\_queue.h](#)

## 4.679 std::random\_access\_iterator\_tag Struct Reference

Inheritance diagram for std::random\_access\_iterator\_tag:



### 4.679.1 Detailed Description

Random-access iterators support a superset of bidirectional iterator operations.

Definition at line 103 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 4.680 std::random\_device Class Reference

### Public Types

- typedef unsigned int [result\\_type](#)

### Public Member Functions

- **random\_device** (const [std::string](#) &\_\_token="mt19937")
- **random\_device** (const [random\\_device](#) &)=delete
- double **entropy** () const noexcept
- [result\\_type](#) **operator**() ()
- void **operator=** (const [random\\_device](#) &)=delete

### Static Public Member Functions

- static constexpr [result\\_type](#) **max** ()
- static constexpr [result\\_type](#) **min** ()

#### 4.680.1 Detailed Description

A standard interface to a platform-specific non-deterministic random number generator (if any are available).

Definition at line 1575 of file random.h.

#### 4.680.2 Member Typedef Documentation

##### 4.680.2.1 typedef unsigned int `std::random_device::result_type`

The type of the generated random value.

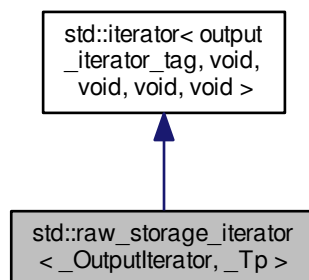
Definition at line 1579 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

#### 4.681 `std::raw_storage_iterator< _OutputIterator, _Tp >` Class Template Reference

Inheritance diagram for `std::raw_storage_iterator< _OutputIterator, _Tp >`:



### Public Types

- typedef void [difference\\_type](#)
- typedef [output\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value\\_type](#)

## Public Member Functions

- [raw\\_storage\\_iterator](#) (\_OutputIterator \_\_x)
- [raw\\_storage\\_iterator](#) & **operator\*** ()
- [raw\\_storage\\_iterator](#)< \_OutputIterator, \_Tp > & **operator++** ()
- [raw\\_storage\\_iterator](#)< \_OutputIterator, \_Tp > **operator++** (int)
- [raw\\_storage\\_iterator](#) & **operator=** (const \_Tp &\_\_element)

## Protected Attributes

- \_OutputIterator **\_M\_iter**

## 4.681.1 Detailed Description

```
template<class _OutputIterator, class _Tp>class std::raw_storage_iterator< _OutputIterator, _Tp >
```

This iterator class lets algorithms store their results into uninitialized memory.

Definition at line 68 of file `stl_raw_storage_iter.h`.

## 4.681.2 Member Typedef Documentation

4.681.2.1 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::difference_type` `[inherited]`

Distance between iterators is represented as this type.

Definition at line 125 of file `stl_iterator_base_types.h`.

4.681.2.2 `typedef output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >::iterator_category` `[inherited]`

One of the [tag types](#).

Definition at line 121 of file `stl_iterator_base_types.h`.

4.681.2.3 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer` `[inherited]`

This type represents a pointer-to-value\_type.

Definition at line 127 of file `stl_iterator_base_types.h`.

4.681.2.4 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference` `[inherited]`

This type represents a reference-to-value\_type.

Definition at line 129 of file `stl_iterator_base_types.h`.

4.681.2.5 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type` `[inherited]`

The type "pointed to" by the iterator.

Definition at line 123 of file `stl_iterator_base_types.h`.

The documentation for this class was generated from the following file:

- [stl\\_raw\\_storage\\_iter.h](#)

## 4.682 `std::regex_error` Class Reference

Inherits `runtime_error`.

### Public Member Functions

- [regex\\_error](#) ([regex\\_constants::error\\_type](#) \_\_ecode)
- [regex\\_constants::error\\_type](#) code () const

### 4.682.1 Detailed Description

A regular expression exception class.

The regular expression library throws objects of this class on error.

Definition at line 136 of file `regex_error.h`.

### 4.682.2 Constructor & Destructor Documentation

#### 4.682.2.1 `std::regex_error::regex_error ( regex_constants::error_type __ecode )` `[explicit]`

Constructs a `regex_error` object.

#### Parameters

<code>__ecode</code>	the <code>regex</code> error code.
----------------------	------------------------------------

### 4.682.3 Member Function Documentation

#### 4.682.3.1 `regex_constants::error_type` `std::regex_error::code ( )` const `[inline]`

Gets the `regex` error code.

#### Returns

the `regex` error code.

Definition at line 157 of file `regex_error.h`.

The documentation for this class was generated from the following file:

- [regex\\_error.h](#)

## 4.683 `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >` Class Template Reference

### Public Types

- typedef `std::ptrdiff_t` **difference\_type**
- typedef [std::forward\\_iterator\\_tag](#) **iterator\_category**
- typedef const [value\\_type](#) \* **pointer**
- typedef const [value\\_type](#) & **reference**
- typedef [basic\\_regex](#)< `_Ch_type`, `_Rx_traits` > **regex\_type**
- typedef [match\\_results](#)< `_Bi_iter` > **value\_type**

## Public Member Functions

- `regex_iterator` ()
- `regex_iterator` ( \_Bi\_iter \_\_a, \_Bi\_iter \_\_b, const `regex_type` &\_\_re, `regex_constants::match_flag_type` \_\_m↵  
m=`regex_constants::match_default`)
- `regex_iterator` (const `regex_iterator` &\_\_rhs)
- `bool operator!=` (const `regex_iterator` &\_\_rhs)
- `const value_type & operator*` ()
- `regex_iterator & operator++` ()
- `regex_iterator operator++` (int)
- `const value_type * operator->` ()
- `regex_iterator & operator=` (const `regex_iterator` &\_\_rhs)
- `bool operator==` (const `regex_iterator` &\_\_rhs)

## 4.683.1 Detailed Description

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> class std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >
```

An iterator adaptor that will provide repeated calls of `regex_search` over a range until no more matches remain.

Definition at line 2204 of file `regex.h`.

## 4.683.2 Constructor &amp; Destructor Documentation

```
4.683.2.1 template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator ( )
```

Provides a singular iterator, useful for indicating one-past-the-end of a range.

**Todo** Implement this function.

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

```
4.683.2.2 template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator ( _Bi_iter __a, _Bi_iter __b, const regex_type & __re, regex_constants::match_flag_type __m = regex_constants::match_default )
```

Constructs a `regex_iterator`...

## Parameters

<code>__a</code>	[IN] The start of a text range to search.
<code>__b</code>	[IN] One-past-the-end of the text range to search.
<code>__re</code>	[IN] The regular expression to match.
<code>__m</code>	[IN] Policy flags for match rules.

**Todo** Implement this function.

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

4.683.2.3 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_iterator ( const regex_iterator<_Bi_iter, _Ch_type, _Rx_traits> & __rhs )`

Copy constructs a `regex_iterator`.

**Todo** Implement this function.

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

#### 4.683.3 Member Function Documentation

4.683.3.1 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> bool std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator!= ( const regex_iterator<_Bi_iter, _Ch_type, _Rx_traits> & __rhs )`

**Todo** Implement this function.

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

4.683.3.2 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> const value_type& std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator* ( )`

**Todo** Implement this function.

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

4.683.3.3 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> regex_iterator& std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator++ ( )`

**Todo** Implement this function.

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

4.683.3.4 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> regex_iterator std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator++ ( int )`

**Todo** Implement this function.

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

4.683.3.5 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> const value_type* std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator-> ( )`

**Todo** Implement this function.

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

```
4.683.3.6 template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits
= regex_traits<_Ch_type>> regex_iterator& std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator= (
const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs )
```

**Todo** Implement this function.

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

```
4.683.3.7 template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits
= regex_traits<_Ch_type>> bool std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator== ( const
regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs )
```

**Todo** Implement this function.

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

The documentation for this class was generated from the following file:

- [regex.h](#)

## 4.684 `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >` Class Template Reference

### Public Types

- typedef `std::ptrdiff_t` **difference\_type**
- typedef `std::forward_iterator_tag` **iterator\_category**
- typedef const `value_type` \* **pointer**
- typedef const `value_type` & **reference**
- typedef `basic_regex< _Ch_type, _Rx_traits >` **regex\_type**
- typedef `sub_match< _Bi_iter >` **value\_type**

### Public Member Functions

- `regex_token_iterator` ()
- `regex_token_iterator` ( `_Bi_iter` \_\_a, `_Bi_iter` \_\_b, const `regex_type` &\_\_re, int \_\_submatch=0, `regex_constants↵::match_flag_type` \_\_m=`regex_constants::match_default`)
- `regex_token_iterator` ( `_Bi_iter` \_\_a, `_Bi_iter` \_\_b, const `regex_type` &\_\_re, const `std::vector< int >` &\_\_↵submatches, `regex_constants::match_flag_type` \_\_m=`regex_constants::match_default`)
- template<`std::size_t` \_\_Nm> `regex_token_iterator` ( `_Bi_iter` \_\_a, `_Bi_iter` \_\_b, const `regex_type` &\_\_re, const int(&\_\_↵submatches)[\_\_Nm], `regex_constants::match_flag_type` \_\_m=`regex_constants::match_default`)
- `regex_token_iterator` (const `regex_token_iterator` &\_\_rhs)
- bool `operator!=` (const `regex_token_iterator` &\_\_rhs)
- const `value_type` & `operator*` ()
- `regex_token_iterator` & `operator++` ()
- `regex_token_iterator` `operator++` (int)
- const `value_type` \* `operator->` ()
- `regex_token_iterator` & `operator=` (const `regex_token_iterator` &\_\_rhs)
- bool `operator==` (const `regex_token_iterator` &\_\_rhs)



## 4.684.1 Detailed Description

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>>>class std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >
```

Iterates over submatches in a range (or *splits* a text string).

The purpose of this iterator is to enumerate all, or all specified, matches of a regular expression within a text range. The dereferenced value of an iterator of this class is a `std::sub_match` object.

Definition at line 2318 of file `regex.h`.

## 4.684.2 Constructor &amp; Destructor Documentation

```
4.684.2.1 template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>>> std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator ( )
```

Default constructs a `regex_token_iterator`.

**Todo** Implement this function.

A default-constructed `regex_token_iterator` is a singular iterator that will compare equal to the one-past-the-end value for any iterator of the same type.

```
4.684.2.2 template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>>> std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator ( _Bi_iter __a, _Bi_iter __b, const regex_type & __re, int __submatch = 0, regex_constants::match_flag_type __m = regex_constants::match_default )
```

Constructs a `regex_token_iterator`...

## Parameters

<code>__a</code>	[IN] The start of the text to search.
<code>__b</code>	[IN] One-past-the-end of the text to search.
<code>__re</code>	[IN] The regular expression to search for.
<code>__submatch</code>	[IN] Which submatch to return. There are some special values for this parameter: <ul style="list-style-type: none"> <li>• -1 each enumerated subexpression does NOT match the regular expression (aka field splitting)</li> <li>• 0 the entire string matching the subexpression is returned for each match within the text.</li> <li>• &gt;0 enumerates only the indicated subexpression from a match within the text.</li> </ul>

<code>__m</code>	[IN] Policy flags for match rules.
------------------	------------------------------------

**Todo** Implement this function.

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

```
4.684.2.3 template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits
= regex_traits<_Ch_type>> std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator
( _Bi_iter __a, _Bi_iter __b, const regex_type & __re, const std::vector< int > & __submatches,
regex_constants::match_flag_type __m = regex_constants::match_default )
```

Constructs a regex\_token\_iterator...

Parameters

<code>__a</code>	[IN] The start of the text to search.
<code>__b</code>	[IN] One-past-the-end of the text to search.
<code>__re</code>	[IN] The regular expression to search for.
<code>__submatches</code>	[IN] A list of subexpressions to return for each regular expression match within the text.
<code>__m</code>	[IN] Policy flags for match rules.

**Todo** Implement this function.

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

```
4.684.2.4 template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits
= regex_traits<_Ch_type>> template<std::size_t _Nm> std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits
>::regex_token_iterator ( _Bi_iter __a, _Bi_iter __b, const regex_type & __re, const int(&) __submatches[ _Nm],
regex_constants::match_flag_type __m = regex_constants::match_default )
```

Constructs a regex\_token\_iterator...

Parameters

<code>__a</code>	[IN] The start of the text to search.
<code>__b</code>	[IN] One-past-the-end of the text to search.
<code>__re</code>	[IN] The regular expression to search for.
<code>__submatches</code>	[IN] A list of subexpressions to return for each regular expression match within the text.
<code>__m</code>	[IN] Policy flags for match rules.

**Todo** Implement this function.

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

```
4.684.2.5 template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits
= regex_traits<_Ch_type>> std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator
( const regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs )
```

Copy constructs a regex\_token\_iterator.

## Parameters

<code>__rhs</code>	[IN] A <code>regex_token_iterator</code> to copy.
--------------------	---

**Todo** Implement this function.

## 4.684.3 Member Function Documentation

4.684.3.1 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> bool std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator!=( const regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits> & __rhs )`

Compares a `regex_token_iterator` to another for inequality.

**Todo** Implement this function.

4.684.3.2 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> const value_type& std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator* ( )`

Dereferences a `regex_token_iterator`.

**Todo** Implement this function.

4.684.3.3 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> regex_token_iterator& std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator++ ( )`

Increments a `regex_token_iterator`.

**Todo** Implement this function.

4.684.3.4 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> regex_token_iterator std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator++ ( int )`

Postincrements a `regex_token_iterator`.

**Todo** Implement this function.

4.684.3.5 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> const value_type* std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator-> ( )`

Selects a `regex_token_iterator` member.

**Todo** Implement this function.

4.684.3.6 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> regex_token_iterator& std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator= ( const regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits> & __rhs )`

Assigns a `regex_token_iterator` to another.

## Parameters

<code>__rhs</code>	[IN] A <code>regex_token_iterator</code> to copy.
--------------------	---

**Todo** Implement this function.

```
4.684.3.7 template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits
= regex_traits<_Ch_type>> bool std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator==( const
regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits> & __rhs )
```

Compares a `regex_token_iterator` to another for equality.

**Todo** Implement this function.

The documentation for this class was generated from the following file:

- [regex.h](#)

## 4.685 std::regex\_traits&lt;\_Ch\_type&gt; Struct Template Reference

## Public Types

- typedef std::ctype\_base::mask **char\_class\_type**
- typedef `_Ch_type` **char\_type**
- typedef [std::locale](#) **locale\_type**
- typedef [std::basic\\_string](#)< `char_type` > **string\_type**

## Public Member Functions

- [regex\\_traits](#) ()
- [locale\\_type](#) [getloc](#) () const
- [locale\\_type](#) [imbue](#) ([locale\\_type](#) \_\_loc)
- bool [isctype](#) (`_Ch_type` \_\_c, `char_class_type` \_\_f) const
- template<typename `_Fwd_iter` > `char_class_type` [lookup\\_classname](#) (`_Fwd_iter` \_\_first, `_Fwd_iter` \_\_last, bool \_\_↔  
icase=false) const
- template<typename `_Fwd_iter` > [string\\_type](#) [lookup\\_collatename](#) (`_Fwd_iter` \_\_first, `_Fwd_iter` \_\_last) const
- template<typename `_Fwd_iter` > [string\\_type](#) [transform](#) (`_Fwd_iter` \_\_first, `_Fwd_iter` \_\_last) const
- template<typename `_Fwd_iter` > [string\\_type](#) [transform\\_primary](#) (`_Fwd_iter` \_\_first, `_Fwd_iter` \_\_last) const
- `char_type` [translate](#) (`char_type` \_\_c) const
- `char_type` [translate\\_nocase](#) (`char_type` \_\_c) const
- int [value](#) (`_Ch_type` \_\_ch, int \_\_radix) const

## Static Public Member Functions

- static std::size\_t [length](#) (const `char_type` \*\_\_p)

## Protected Attributes

- [locale\\_type](#) **\_M\_locale**

#### 4.685.1 Detailed Description

```
template<typename _Ch_type> struct std::regex_traits< _Ch_type >
```

Class `regex_traits`. Describes aspects of a regular expression.

A regular expression traits class that satisfies the requirements of section [28.7].

The class `regex` is parameterized around a set of related types and functions used to complete the definition of its semantics. This class satisfies the requirements of such a traits class.

Definition at line 51 of file `regex.h`.

#### 4.685.2 Constructor & Destructor Documentation

4.685.2.1 `template<typename _Ch_type > std::regex_traits< _Ch_type >::regex_traits ( ) [inline]`

Constructs a default traits object.

Definition at line 63 of file `regex.h`.

#### 4.685.3 Member Function Documentation

4.685.3.1 `template<typename _Ch_type > locale_type std::regex_traits< _Ch_type >::getloc ( ) const [inline]`

Gets a copy of the current locale in use by the `regex_traits` object.

Definition at line 270 of file `regex.h`.

4.685.3.2 `template<typename _Ch_type > locale_type std::regex_traits< _Ch_type >::imbue ( locale_type __loc ) [inline]`

Imbues the `regex_traits` object with a copy of a new locale.

##### Parameters

<code>__loc</code>	A locale.
--------------------	-----------

##### Returns

a copy of the previous locale in use by the `regex_traits` object.

##### Note

Calling `imbue` with a different locale than the one currently in use invalidates all cached data held by `*this`.

Definition at line 259 of file `regex.h`.

References `std::swap()`.

4.685.3.3 `template<typename _Ch_type > static std::size_t std::regex_traits< _Ch_type >::length ( const char_type * __p ) [inline], [static]`

Gives the length of a C-style string starting at `__p`.

## Parameters

<code>__p</code>	a pointer to the start of a character sequence.
------------------	---

## Returns

the number of characters between `*__p` and the first default-initialized value of type `char_type`. In other words, uses the C-string algorithm for determining the length of a sequence of characters.

Definition at line 76 of file `regex.h`.

4.685.3.4 `template<typename _Ch_type> template<typename _Fwd_iter> char_class_type std::regex_traits<_Ch_type>::lookup_classname ( _Fwd_iter __first, _Fwd_iter __last, bool __icase = false ) const [inline]`

Maps one or more characters to a named character classification.

## Parameters

<code>__first</code>	beginning of the character sequence.
<code>__last</code>	one-past-the-end of the character sequence.
<code>__icase</code>	ignores the case of the classification name.

## Returns

an unspecified value that represents the character classification named by the character sequence designated by the iterator range `[__first, __last)`. If `icase` is true, the returned mask identifies the classification regardless of the case of the characters to be matched (for example, `[[:lower:]]` is the same as `[[:alpha:]]`), otherwise a case-dependent classification is returned. The value returned shall be independent of the case of the characters in the character sequence. If the name is not recognized then returns a value that compares equal to 0.

At least the following names (or their wide-character equivalent) are supported.

- `d`
- `w`
- `s`
- `alnum`
- `alpha`
- `blank`
- `cntrl`
- `digit`
- `graph`
- `lower`
- `print`
- `punct`
- `space`
- `upper`

- `xdigit`

**Todo** Implement this function.

Definition at line 215 of file `regex.h`.

4.685.3.5 `template<typename _Ch_type > template<typename _Fwd_iter > string_type std::regex_traits< _Ch_type >::lookup_collatename ( _Fwd_iter __first, _Fwd_iter __last ) const [inline]`

Gets a collation element by name.

#### Parameters

<code>__first</code>	beginning of the collation element name.
<code>__last</code>	one-past-the-end of the collation element name.

#### Returns

a sequence of one or more characters that represents the collating element consisting of the character sequence designated by the iterator range `[__first, __last)`. Returns an empty string if the character sequence is not a valid collating element.

**Todo** Implement this function.

Definition at line 171 of file `regex.h`.

4.685.3.6 `template<typename _Ch_type > template<typename _Fwd_iter > string_type std::regex_traits< _Ch_type >::transform ( _Fwd_iter __first, _Fwd_iter __last ) const [inline]`

Gets a sort key for a character sequence.

#### Parameters

<code>__first</code>	beginning of the character sequence.
<code>__last</code>	one-past-the-end of the character sequence.

Returns a sort key for the character sequence designated by the iterator range `[F1, F2)` such that if the character sequence `[G1, G2)` sorts before the character sequence `[H1, H2)` then `v.transform(G1, G2) < v.transform(H1, H2)`.

What this really does is provide a more efficient way to compare a string to multiple other strings in locales with fancy collation rules and equivalence classes.

#### Returns

a locale-specific sort key equivalent to the input range.

#### Exceptions

<code>std::bad_cast</code>	if the current locale does not have a collate facet.
----------------------------	--

Definition at line 129 of file `regex.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::data()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

4.685.3.7 `template<typename _Ch_type > template<typename _Fwd_iter > string_type std::regex_traits< _Ch_type >::transform_primary ( _Fwd_iter __first, _Fwd_iter __last ) const [inline]`

Gets a sort key for a character sequence, independent of case.

## Parameters

<code>__first</code>	beginning of the character sequence.
<code>__last</code>	one-past-the-end of the character sequence.

Effects: if `typeid(use_facet<collate<_Ch_type>>) == typeid(collate_byname<_Ch_type>)` and the form of the sort key returned by `collate_byname<_Ch_type>::transform(__first, __last)` is known and can be converted into a primary sort key then returns that key, otherwise returns an empty string.

**Todo** Implement this function.

Definition at line 153 of file `regex.h`.

```
4.685.3.8 template<typename _Ch_type > char_type std::regex_traits<_Ch_type>::translate ( char_type __c ) const
[inline]
```

Performs the identity translation.

## Parameters

<code>__c</code>	A character to the locale-specific character set.
------------------	---

## Returns

`__c`.

Definition at line 87 of file `regex.h`.

```
4.685.3.9 template<typename _Ch_type > char_type std::regex_traits<_Ch_type>::translate_nocase ( char_type __c ) const
[inline]
```

Translates a character into a case-insensitive equivalent.

## Parameters

<code>__c</code>	A character to the locale-specific character set.
------------------	---

## Returns

the locale-specific lower-case equivalent of `__c`.

## Exceptions

<code>std::bad_cast</code>	if the imbued locale does not support the ctype facet.
----------------------------	--

Definition at line 100 of file `regex.h`.

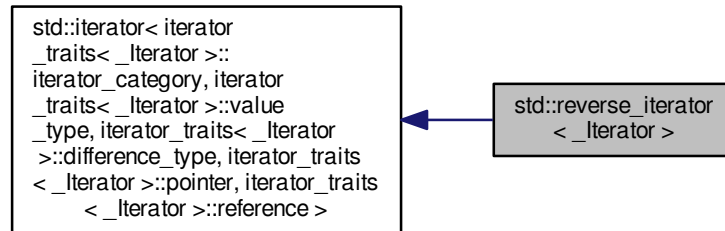
The documentation for this struct was generated from the following file:

- [regex.h](#)



#### 4.686 `std::reverse_iterator<_Iterator>` Class Template Reference

Inheritance diagram for `std::reverse_iterator<_Iterator>`:



#### Public Types

- typedef `__traits_type::difference_type` **difference\_type**
- typedef `iterator_traits<_Iterator>::iterator_category` **iterator\_category**
- typedef `_Iterator` **iterator\_type**
- typedef `__traits_type::pointer` **pointer**
- typedef `__traits_type::reference` **reference**
- typedef `iterator_traits<_Iterator>::value_type` **value\_type**

#### Public Member Functions

- `reverse_iterator()`
- `reverse_iterator(iterator_type __x)`
- `reverse_iterator(const reverse_iterator &__x)`
- `template<typename _Iter> reverse_iterator(const reverse_iterator<_Iter> &__x)`
- `iterator_type base() const`
- `reference operator*() const`
- `reverse_iterator operator+ (difference_type __n) const`
- `reverse_iterator & operator++ ()`
- `reverse_iterator operator++ (int)`
- `reverse_iterator & operator+= (difference_type __n)`
- `reverse_iterator operator- (difference_type __n) const`
- `reverse_iterator & operator-- ()`
- `reverse_iterator operator-- (int)`
- `reverse_iterator & operator-= (difference_type __n)`
- `pointer operator-> () const`
- `reference operator[] (difference_type __n) const`

#### Protected Types

- typedef `iterator_traits<_Iterator> __traits_type`

## Protected Attributes

- `_Iterator current`

## 4.686.1 Detailed Description

```
template<typename _Iterator> class std::reverse_iterator< _Iterator >
```

Bidirectional and random access iterators have corresponding reverse iterator adaptors that iterate through the data structure in the opposite direction. They have the same signatures as the corresponding iterators. The fundamental relation between a reverse iterator and its corresponding iterator `i` is established by the identity:

```
&*(reverse_iterator(i)) == &(i - 1)
```

*This mapping is dictated by the fact that while there is always a pointer past the end of an array, there might not be a valid pointer before the beginning of an array. [24.4.1]/1,2*

Reverse iterators can be tricky and surprising at first. Their semantics make sense, however, and the trickiness is a side effect of the requirement that the iterators must be safe.

Definition at line 96 of file `stl_iterator.h`.

## 4.686.2 Member Typedef Documentation

```
4.686.2.1 typedef iterator_traits< _Iterator >::iterator_category std::iterator< iterator_traits< _Iterator
>::iterator_category, iterator_traits< _Iterator >::value_type, iterator_traits< _Iterator >::difference_type,
iterator_traits< _Iterator >::pointer, iterator_traits< _Iterator >::reference >::iterator_category [inherited]
```

One of the [tag types](#).

Definition at line 121 of file `stl_iterator_base_types.h`.

```
4.686.2.2 typedef iterator_traits< _Iterator >::value_type std::iterator< iterator_traits< _Iterator >::iterator_category
, iterator_traits< _Iterator >::value_type, iterator_traits< _Iterator >::difference_type, iterator_traits< _Iterator
>::pointer, iterator_traits< _Iterator >::reference >::value_type [inherited]
```

The type "pointed to" by the iterator.

Definition at line 123 of file `stl_iterator_base_types.h`.

## 4.686.3 Constructor &amp; Destructor Documentation

```
4.686.3.1 template<typename _Iterator> std::reverse_iterator< _Iterator >::reverse_iterator ( ) [inline]
```

The default constructor value-initializes member `current`. If it is a pointer, that means it is zero-initialized.

Definition at line 120 of file `stl_iterator.h`.

Referenced by `std::reverse_iterator< _Iterator >::operator+()`, and `std::reverse_iterator< _Iterator >::operator-()`.

```
4.686.3.2 template<typename _Iterator> std::reverse_iterator< _Iterator >::reverse_iterator ( iterator_type __x )
[inline], [explicit]
```

This iterator will move in the opposite direction that `x` does.

Definition at line 126 of file `stl_iterator.h`.

4.686.3.3 `template<typename _Iterator> std::reverse_iterator<_Iterator>::reverse_iterator ( const reverse_iterator<_Iterator> &__x ) [inline]`

The copy constructor is normal.

Definition at line 131 of file `stl_iterator.h`.

4.686.3.4 `template<typename _Iterator> template<typename _Iter> std::reverse_iterator<_Iterator>::reverse_iterator ( const reverse_iterator<_Iter> &__x ) [inline]`

A `reverse_iterator` across other types can be copied if the underlying iterator can be converted to the type of `current`.

Definition at line 139 of file `stl_iterator.h`.

#### 4.686.4 Member Function Documentation

4.686.4.1 `template<typename _Iterator> iterator_type std::reverse_iterator<_Iterator>::base ( ) const [inline]`

##### Returns

`current`, the iterator used for underlying work.

Definition at line 146 of file `stl_iterator.h`.

Referenced by `std::operator==()`.

4.686.4.2 `template<typename _Iterator> reference std::reverse_iterator<_Iterator>::operator* ( ) const [inline]`

##### Returns

A reference to the value at `-current`

This requires that `-current` is dereferenceable.

##### Warning

This implementation requires that for an iterator of the underlying iterator type, `x`, a reference obtained by `*x` remains valid after `x` has been modified or destroyed. This is a bug: <http://gcc.gnu.org/PR51823>

Definition at line 160 of file `stl_iterator.h`.

Referenced by `std::reverse_iterator<_Iterator>::operator->()`.

4.686.4.3 `template<typename _Iterator> reverse_iterator std::reverse_iterator<_Iterator>::operator+ ( difference_type __n ) const [inline]`

##### Returns

A `reverse_iterator` that refers to `current - __n`

The underlying iterator must be a Random Access Iterator.

Definition at line 231 of file `stl_iterator.h`.

References `std::reverse_iterator<_Iterator>::reverse_iterator()`.

4.686.4.4 `template<typename _Iterator> reverse_iterator& std::reverse_iterator<_Iterator>::operator++ ( ) [inline]`

**Returns**

\*this

Decrements the underlying iterator.

Definition at line 181 of file stl\_iterator.h.

**4.686.4.5** `template<typename _Iterator> reverse_iterator std::reverse_iterator<_Iterator>::operator++ ( int )`  
`[inline]`

**Returns**

The original value of \*this

Decrements the underlying iterator.

Definition at line 193 of file stl\_iterator.h.

**4.686.4.6** `template<typename _Iterator> reverse_iterator& std::reverse_iterator<_Iterator>::operator+= ( difference_type __n )` `[inline]`

**Returns**

\*this

Moves the underlying iterator backwards \_\_n steps. The underlying iterator must be a Random Access Iterator.

Definition at line 241 of file stl\_iterator.h.

**4.686.4.7** `template<typename _Iterator> reverse_iterator std::reverse_iterator<_Iterator>::operator- ( difference_type __n ) const` `[inline]`

**Returns**

A reverse\_iterator that refers to current - \_\_n

The underlying iterator must be a Random Access Iterator.

Definition at line 253 of file stl\_iterator.h.

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

**4.686.4.8** `template<typename _Iterator> reverse_iterator& std::reverse_iterator<_Iterator>::operator-- ( )`  
`[inline]`

**Returns**

\*this

Increments the underlying iterator.

Definition at line 206 of file stl\_iterator.h.

**4.686.4.9** `template<typename _Iterator> reverse_iterator std::reverse_iterator<_Iterator>::operator-- ( int )`  
`[inline]`

**Returns**

A reverse\_iterator with the previous value of \*this

Increments the underlying iterator.

Definition at line 218 of file stl\_iterator.h.

4.686.4.10 `template<typename _Iterator> reverse_iterator& std::reverse_iterator<_Iterator>::operator=(  
difference_type __n) [inline]`

#### Returns

\*this

Moves the underlying iterator forwards `__n` steps. The underlying iterator must be a Random Access Iterator.

Definition at line 263 of file `stl_iterator.h`.

4.686.4.11 `template<typename _Iterator> pointer std::reverse_iterator<_Iterator>::operator->() const [inline]`

#### Returns

A pointer to the value at `-current`

This requires that `-current` is dereferenceable.

Definition at line 172 of file `stl_iterator.h`.

References `std::reverse_iterator<_Iterator>::operator*()`.

4.686.4.12 `template<typename _Iterator> reference std::reverse_iterator<_Iterator>::operator[]( difference_type __n )  
const [inline]`

#### Returns

The value at `current - __n - 1`

The underlying iterator must be a Random Access Iterator.

Definition at line 275 of file `stl_iterator.h`.

The documentation for this class was generated from the following file:

- [stl\\_iterator.h](#)

## 4.687 std::seed\_seq Class Reference

### Public Types

- `typedef uint_least32_t result_type`

### Public Member Functions

- [seed\\_seq\(\)](#)
- `template<typename _IntType> seed_seq (std::initializer_list<_IntType> il)`
- `template<typename _InputIterator> seed_seq (_InputIterator __begin, _InputIterator __end)`
- `template<typename _RandomAccessIterator> void generate (_RandomAccessIterator __begin, _RandomAccessIterator __end)`
- `template<typename OutputIterator> void param (OutputIterator __dest) const`
- `size_t size () const`

### 4.687.1 Detailed Description

The `seed_seq` class generates sequences of seeds for random number generators.

Definition at line 6025 of file `random.h`.

## 4.687.2 Member Typedef Documentation

## 4.687.2.1 typedef uint\_least32\_t std::seed\_seq::result\_type

The type of the seed vales.

Definition at line 6030 of file random.h.

## 4.687.3 Constructor &amp; Destructor Documentation

## 4.687.3.1 std::seed\_seq::seed\_seq( ) [inline]

Default constructor.

Definition at line 6033 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

## 4.688 std::set&lt; \_Key, \_Compare, \_Alloc &gt; Class Template Reference

## Public Types

- typedef \_Key [key\\_type](#)
- typedef \_Key [value\\_type](#)
- typedef \_Compare [key\\_compare](#)
- typedef \_Compare [value\\_compare](#)
- typedef \_Alloc [allocator\\_type](#)
- typedef \_Key\_alloc\_type::pointer [pointer](#)
- typedef \_Key\_alloc\_type::const\_pointer [const\\_pointer](#)
- typedef \_Key\_alloc\_type::reference [reference](#)
- typedef \_Key\_alloc\_type::const\_reference [const\\_reference](#)
- typedef \_Rep\_type::const\_iterator [iterator](#)
- typedef \_Rep\_type::const\_iterator [const\\_iterator](#)
- typedef \_Rep\_type::const\_reverse\_iterator [reverse\\_iterator](#)
- typedef \_Rep\_type::const\_reverse\_iterator [const\\_reverse\\_iterator](#)
- typedef \_Rep\_type::size\_type [size\\_type](#)
- typedef \_Rep\_type::difference\_type [difference\\_type](#)

## Public Member Functions

- [set](#) ()
- [set](#) (const \_Compare &\_\_comp, const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- template<typename \_InputIterator > [set](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_InputIterator > [set](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_comp, const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [set](#) (const [set](#) &\_\_x)
- [set](#) ([set](#) &&\_\_x) noexcept(is\_nothrow\_copy\_constructible< \_Compare >::value)
- [set](#) (initializer\_list< [value\\_type](#) > \_\_l, const \_Compare &\_\_comp=\_Compare(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())

- `iterator begin` () const noexcept
  - `iterator cbegin` () const noexcept
  - `iterator cend` () const noexcept
  - `void clear` () noexcept
  - `size_type count` (const `key_type` &\_\_x) const
  - `reverse_iterator crbegin` () const noexcept
  - `reverse_iterator crend` () const noexcept
  - `template<typename... _Args> std::pair< iterator, bool > emplace` (\_Args &&... \_\_args)
  - `template<typename... _Args> iterator emplace_hint` (const\_iterator \_\_pos, \_Args &&... \_\_args)
  - `bool empty` () const noexcept
  - `iterator end` () const noexcept
  - `_GLIBCXX_ABI_TAG_CXX11 iterator erase` (const\_iterator \_\_position)
  - `size_type erase` (const `key_type` &\_\_x)
  - `_GLIBCXX_ABI_TAG_CXX11 iterator erase` (const\_iterator \_\_first, const\_iterator \_\_last)
  - `allocator_type get_allocator` () const noexcept
  - `std::pair< iterator, bool > insert` (const `value_type` &\_\_x)
  - `std::pair< iterator, bool > insert` (`value_type` &&\_\_x)
  - `iterator insert` (const\_iterator \_\_position, const `value_type` &\_\_x)
  - `iterator insert` (const\_iterator \_\_position, `value_type` &&\_\_x)
  - `template<typename _InputIterator > void insert` (\_InputIterator \_\_first, \_InputIterator \_\_last)
  - `void insert` (initializer\_list< `value_type` > \_\_l)
  - `key_compare key_comp` () const
  - `size_type max_size` () const noexcept
  - `set & operator=` (const `set` &\_\_x)
  - `set & operator=` (`set` &&\_\_x)
  - `set & operator=` (initializer\_list< `value_type` > \_\_l)
  - `reverse_iterator rbegin` () const noexcept
  - `reverse_iterator rend` () const noexcept
  - `size_type size` () const noexcept
  - `void swap` (`set` &\_\_x)
  - `value_compare value_comp` () const
- 
- `iterator find` (const `key_type` &\_\_x)
  - `const_iterator find` (const `key_type` &\_\_x) const
- 
- `iterator lower_bound` (const `key_type` &\_\_x)
  - `const_iterator lower_bound` (const `key_type` &\_\_x) const
- 
- `iterator upper_bound` (const `key_type` &\_\_x)
  - `const_iterator upper_bound` (const `key_type` &\_\_x) const
- 
- `std::pair< iterator, iterator > equal_range` (const `key_type` &\_\_x)
  - `std::pair< const_iterator, const_iterator > equal_range` (const `key_type` &\_\_x) const

## Friends

- `template<typename _K1, typename _C1, typename _A1 > bool operator<` (const `set`< \_K1, \_C1, \_A1 > &, const `set`< \_K1, \_C1, \_A1 > &)
- `template<typename _K1, typename _C1, typename _A1 > bool operator==` (const `set`< \_K1, \_C1, \_A1 > &, const `set`< \_K1, \_C1, \_A1 > &)

## 4.688.1 Detailed Description

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> class std::set<_Key,
_Compare, _Alloc>
```

A standard container made up of unique keys, which can be retrieved in logarithmic time.



## Template Parameters

<code>_Key</code>	Type of key objects.
<code>_Compare</code>	Comparison function object type, defaults to <code>less&lt;_Key&gt;</code> .
<code>_Alloc</code>	Allocator type, defaults to <code>allocator&lt;_Key&gt;</code> .

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using unique keys).

Sets support bidirectional iterators.

The private tree data is declared exactly the same way for set and multiset; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 90 of file `stl_set.h`.

## 4.688.2 Member Typedef Documentation

4.688.2.1 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Alloc std::set<_Key, _Compare, _Alloc>::allocator_type`

Public typedefs.

Definition at line 107 of file `stl_set.h`.

4.688.2.2 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Rep_type::const_iterator std::set<_Key, _Compare, _Alloc>::const_iterator`

Iterator-related typedefs.

Definition at line 128 of file `stl_set.h`.

4.688.2.3 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Key_alloc_type::const_pointer std::set<_Key, _Compare, _Alloc>::const_pointer`

Iterator-related typedefs.

Definition at line 121 of file `stl_set.h`.

4.688.2.4 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Key_alloc_type::const_reference std::set<_Key, _Compare, _Alloc>::const_reference`

Iterator-related typedefs.

Definition at line 123 of file `stl_set.h`.

4.688.2.5 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Rep_type::const_reverse_iterator std::set<_Key, _Compare, _Alloc>::const_reverse_iterator`

Iterator-related typedefs.

Definition at line 130 of file `stl_set.h`.

4.688.2.6 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Rep_type::difference_type std::set<_Key, _Compare, _Alloc>::difference_type`

Iterator-related typedefs.

Definition at line 132 of file `stl_set.h`.

4.688.2.7 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef  
_Rep_type::const_iterator std::set<_Key, _Compare, _Alloc>::iterator`

Iterator-related typedefs.

Definition at line 127 of file stl\_set.h.

4.688.2.8 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef  
_Compare std::set<_Key, _Compare, _Alloc>::key_compare`

Public typedefs.

Definition at line 105 of file stl\_set.h.

4.688.2.9 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef  
_Key std::set<_Key, _Compare, _Alloc>::key_type`

Public typedefs.

Definition at line 103 of file stl\_set.h.

4.688.2.10 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef  
_Key_alloc_type::pointer std::set<_Key, _Compare, _Alloc>::pointer`

Iterator-related typedefs.

Definition at line 120 of file stl\_set.h.

4.688.2.11 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef  
_Key_alloc_type::reference std::set<_Key, _Compare, _Alloc>::reference`

Iterator-related typedefs.

Definition at line 122 of file stl\_set.h.

4.688.2.12 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef  
_Rep_type::const_reverse_iterator std::set<_Key, _Compare, _Alloc>::reverse_iterator`

Iterator-related typedefs.

Definition at line 129 of file stl\_set.h.

4.688.2.13 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef  
_Rep_type::size_type std::set<_Key, _Compare, _Alloc>::size_type`

Iterator-related typedefs.

Definition at line 131 of file stl\_set.h.

4.688.2.14 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef  
_Compare std::set<_Key, _Compare, _Alloc>::value_compare`

Public typedefs.

Definition at line 106 of file stl\_set.h.

4.688.2.15 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef  
_Key std::set<_Key, _Compare, _Alloc>::value_type`

Public typedefs.

Definition at line 104 of file `stl_set.h`.

#### 4.688.3 Constructor & Destructor Documentation

4.688.3.1 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>  
std::set<_Key, _Compare, _Alloc>::set ( ) [inline]`

Default constructor creates no elements.

Definition at line 139 of file `stl_set.h`.

4.688.3.2 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>  
std::set<_Key, _Compare, _Alloc>::set ( const _Compare & __comp, const allocator_type & __a =  
allocator_type() ) [inline], [explicit]`

Creates a set with no elements.

Parameters

<code>__comp</code>	Comparator to use.
<code>__a</code>	An allocator object.

Definition at line 148 of file `stl_set.h`.

4.688.3.3 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>  
template<typename _InputIterator> std::set<_Key, _Compare, _Alloc>::set ( _InputIterator __first, _InputIterator  
__last ) [inline]`

Builds a set from a range.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

Create a set consisting of copies of the elements from `[__first, __last)`. This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(`__first, __last`)).

Definition at line 163 of file `stl_set.h`.

4.688.3.4 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>  
template<typename _InputIterator> std::set<_Key, _Compare, _Alloc>::set ( _InputIterator __first, _InputIterator  
__last, const _Compare & __comp, const allocator_type & __a = allocator_type() ) [inline]`

Builds a set from a range.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__comp</code>	A comparison functor.
<code>__a</code>	An allocator object.

Create a set consisting of copies of the elements from `[__first, __last)`. This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(`__first, __last`)).

Definition at line 180 of file `stl_set.h`.

```
4.688.3.5  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
           std::set<_Key, _Compare, _Alloc>::set ( const set<_Key, _Compare, _Alloc> &__x )  [inline]
```

Set copy constructor.

## Parameters

<code>__x</code>	A set of identical element and allocator types.
------------------	---

The newly-created set uses a copy of the allocation object used by `__x`.

Definition at line 193 of file `stl_set.h`.

```
4.688.3.6  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            std::set<_Key, _Compare, _Alloc>::set ( set<_Key, _Compare, _Alloc> && __x )  [inline], [noexcept]
```

Set move constructor

## Parameters

<code>__x</code>	A set of identical element and allocator types.
------------------	---

The newly-created set contains the exact contents of `x`. The contents of `x` are a valid, but unspecified set.

Definition at line 204 of file `stl_set.h`.

```
4.688.3.7  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            std::set<_Key, _Compare, _Alloc>::set ( initializer_list<value_type> & __l, const _Compare & __comp =
            _Compare(), const allocator_type & __a = allocator_type() )  [inline]
```

Builds a set from an `initializer_list`.

## Parameters

<code>__l</code>	An <code>initializer_list</code> .
<code>__comp</code>	A comparison functor.
<code>__a</code>	An allocator object.

Create a set consisting of copies of the elements in the list. This is linear in  $N$  if the list is already sorted, and  $N \log N$  otherwise (where  $N$  is `__l.size()`).

Definition at line 218 of file `stl_set.h`.

## 4.688.4 Member Function Documentation

```
4.688.4.1  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator
            std::set<_Key, _Compare, _Alloc>::begin ( ) const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

Definition at line 298 of file `stl_set.h`.

```
4.688.4.2  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator
            std::set<_Key, _Compare, _Alloc>::cbegin ( ) const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

Definition at line 335 of file `stl_set.h`.

```
4.688.4.3  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator
            std::set<_Key, _Compare, _Alloc>::cend ( ) const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

Definition at line 344 of file stl\_set.h.

4.688.4.4 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> void  
std::set< _Key, _Compare, _Alloc >::clear ( ) [inline], [noexcept]`

Erases all elements in a set. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 630 of file stl\_set.h.

Referenced by `std::set< _StateIdT >::operator=()`.

4.688.4.5 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
size_type std::set< _Key, _Compare, _Alloc >::count ( const key_type & __x ) const [inline]`

Finds the number of elements.

Parameters

<code>__x</code>	Element to located.
------------------	---------------------

Returns

Number of elements with specified key.

This function only makes sense for multisets; for set the result will either be 0 (not present) or 1 (present).

Definition at line 644 of file stl\_set.h.

4.688.4.6 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
reverse_iterator std::set< _Key, _Compare, _Alloc >::rbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

Definition at line 353 of file stl\_set.h.

4.688.4.7 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
reverse_iterator std::set< _Key, _Compare, _Alloc >::crend ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

Definition at line 362 of file stl\_set.h.

4.688.4.8 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
template<typename... _Args> std::pair<iterator, bool> std::set< _Key, _Compare, _Alloc >::emplace ( _Args  
&&... __args ) [inline]`

Attempts to build and insert an element into the set.

Parameters

<code>__args</code>	Arguments used to generate an element.
---------------------	--

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to build and insert an element into the set. A set relies on unique keys and thus an element is only inserted if it is not already present in the set.

Insertion requires logarithmic time.

Definition at line 413 of file `stl_set.h`.

```
4.688.4.9  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
           template<typename... _Args> iterator std::set<_Key, _Compare, _Alloc>::emplace_hint ( const_iterator __pos,
           _Args &&... __args ) [inline]
```

Attempts to insert an element into the set.

**Parameters**

<code>__pos</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__args</code>	Arguments used to generate the element to be inserted.

**Returns**

An iterator that points to the element with key equivalent to the one generated from `__args` (may or may not be the element itself).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `emplace()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html>.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 439 of file `stl_set.h`.

```
4.688.4.10 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> bool
           std::set<_Key, _Compare, _Alloc>::empty ( ) const [inline], [noexcept]
```

Returns true if the set is empty.

Definition at line 368 of file `stl_set.h`.

```
4.688.4.11 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
           iterator std::set<_Key, _Compare, _Alloc>::end ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

Definition at line 307 of file `stl_set.h`.

```
4.688.4.12 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
           std::pair<iterator, iterator> std::set<_Key, _Compare, _Alloc>::equal_range ( const key_type & __x )
           [inline]
```

Finds a subsequence matching given key.

## Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

## Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 724 of file stl\_set.h.

```
4.688.4.13 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            std::pair<const_iterator, const_iterator> std::set<_Key, _Compare, _Alloc >::equal_range ( const
            key_type & __x ) const [inline]
```

Finds a subsequence matching given key.

## Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

## Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 728 of file stl\_set.h.

```
4.688.4.14 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            _GLIBCXX_ABI_TAG_CXX11 iterator std::set<_Key, _Compare, _Alloc >::erase ( const_iterator __position )
            [inline]
```

Erases an element from a set.

## Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

## Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.



This function erases an element, pointed to by the given iterator, from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 551 of file `stl_set.h`.

```
4.688.4.15  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            size_type std::set<_Key, _Compare, _Alloc>::erase ( const key_type & __x )  [inline]
```

Erases elements according to the provided key.

#### Parameters

<code>__x</code>	Key of element to be erased.
------------------	------------------------------

#### Returns

The number of elements erased.

This function erases all the elements located by the given key from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 581 of file `stl_set.h`.

```
4.688.4.16  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            _GLIBCXX_ABI_TAG_CXX11 iterator std::set<_Key, _Compare, _Alloc>::erase ( const_iterator __first,
            const_iterator __last )  [inline]
```

Erases a [`__first`,`__last`) range of elements from a set.

#### Parameters

<code>__first</code>	Iterator pointing to the start of the range to be erased.
<code>__last</code>	Iterator pointing to the end of the range to be erased.

#### Returns

The iterator `__last`.

This function erases a sequence of elements from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 603 of file `stl_set.h`.

```
4.688.4.17  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            iterator std::set<_Key, _Compare, _Alloc>::find ( const key_type & __x )  [inline]
```

Tries to locate an element in a set.

#### Parameters

<code>__x</code>	Element to be located.
------------------	------------------------

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

Definition at line 662 of file stl\_set.h.

```
4.688.4.18 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
    const_iterator std::set< _Key, _Compare, _Alloc >::find ( const key_type & __x ) const    [inline]
```

Tries to locate an element in a set.

**Parameters**

<code>__x</code>	Element to be located.
------------------	------------------------

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

Definition at line 666 of file stl\_set.h.

```
4.688.4.19 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
    allocator_type std::set< _Key, _Compare, _Alloc >::get_allocator ( ) const    [inline], [noexcept]
```

Returns the allocator object with which the set was constructed.

Definition at line 289 of file stl\_set.h.

```
4.688.4.20 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
    std::pair<iterator, bool> std::set< _Key, _Compare, _Alloc >::insert ( const value_type & __x )    [inline]
```

Attempts to insert an element into the set.

**Parameters**

<code>__x</code>	Element to be inserted.
------------------	-------------------------

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the set. A set relies on unique keys and thus an element is only inserted if it is not already present in the set.

Insertion requires logarithmic time.

Definition at line 460 of file stl\_set.h.

Referenced by std::set< \_StateIdT >::insert(), and std::set< \_StateIdT >::operator=().

```
4.688.4.21 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
    iterator std::set< _Key, _Compare, _Alloc >::insert ( const_iterator __position, const value_type & __x )
    [inline]
```

Attempts to insert an element into the set.

## Parameters

<code>__position</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__x</code>	Element to be inserted.

## Returns

An iterator that points to the element with key of `__x` (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html>.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 497 of file `stl_set.h`.

```
4.688.4.22 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _InputIterator > void std::set<_Key, _Compare, _Alloc >::insert ( _InputIterator __first,
_InputIterator __last ) [inline]
```

A template function that attempts to insert a range of elements.

## Parameters

<code>__first</code>	Iterator pointing to the start of the range to be inserted.
<code>__last</code>	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 517 of file `stl_set.h`.

```
4.688.4.23 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> void
std::set<_Key, _Compare, _Alloc >::insert ( initializer_list< value_type > __l ) [inline]
```

Attempts to insert a list of elements into the set.

## Parameters

<code>__l</code>	A <code>std::initializer_list&lt;value_type&gt;</code> of elements to be inserted.
------------------	--

Complexity similar to that of the range constructor.

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

```
4.688.4.24 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
key_compare std::set<_Key, _Compare, _Alloc >::key_comp ( ) const [inline]
```

Returns the comparison object with which the set was constructed.

Definition at line 281 of file `stl_set.h`.

```
4.688.4.25 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::set<_Key, _Compare, _Alloc >::lower_bound ( const key_type & __x ) [inline]
```

Finds the beginning of a subsequence matching given key.

## Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

## Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 683 of file stl\_set.h.

```
4.688.4.26 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
const_iterator std::set<_Key, _Compare, _Alloc>::lower_bound( const key_type & __x ) const [inline]
```

Finds the beginning of a subsequence matching given key.

## Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

## Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 687 of file stl\_set.h.

```
4.688.4.27 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
size_type std::set<_Key, _Compare, _Alloc>::max_size( ) const [inline], [noexcept]
```

Returns the maximum size of the set.

Definition at line 378 of file stl\_set.h.

```
4.688.4.28 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> set&
std::set<_Key, _Compare, _Alloc>::operator= ( const set<_Key, _Compare, _Alloc> & __x ) [inline]
```

Set assignment operator.

## Parameters

<code>__x</code>	A set of identical element and allocator types.
------------------	---

All the elements of `__x` are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 233 of file stl\_set.h.

```
4.688.4.29 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> set&
std::set<_Key, _Compare, _Alloc>::operator= ( set<_Key, _Compare, _Alloc> && __x ) [inline]
```

Set move assignment operator.

## Parameters

<code>__x</code>	A set of identical element and allocator types.
------------------	---

The contents of `__x` are moved into this set (without copying). `__x` is a valid, but unspecified set.

Definition at line 248 of file `stl_set.h`.

**4.688.4.30** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> set& std::set<_Key, _Compare, _Alloc>::operator= ( initializer_list<value_type> __l ) [inline]`

Set list assignment operator.

## Parameters

<code>__l</code>	An <code>initializer_list</code> .
------------------	------------------------------------

This function fills a set with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the set and that the resulting set's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 269 of file `stl_set.h`.

**4.688.4.31** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> reverse_iterator std::set<_Key, _Compare, _Alloc>::rbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

Definition at line 316 of file `stl_set.h`.

**4.688.4.32** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> reverse_iterator std::set<_Key, _Compare, _Alloc>::rend ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

Definition at line 325 of file `stl_set.h`.

**4.688.4.33** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> size_type std::set<_Key, _Compare, _Alloc>::size ( ) const [inline], [noexcept]`

Returns the size of the set.

Definition at line 373 of file `stl_set.h`.

**4.688.4.34** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> void std::set<_Key, _Compare, _Alloc>::swap ( set<_Key, _Compare, _Alloc> & __x ) [inline]`

Swaps data with another set.

## Parameters

<code>__x</code>	A set of the same element and allocator types.
------------------	--

This exchanges the elements between two sets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

Definition at line 393 of file `stl_set.h`.

Referenced by `std::set<_StateIDT>::operator=()`, and `std::swap()`.

```
4.688.4.35  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
            iterator std::set<_Key, _Compare, _Alloc>::upper_bound( const key_type & __x )  [inline]
```

Finds the end of a subsequence matching given key.

## Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

## Returns

Iterator pointing to the first element greater than key, or end().

Definition at line 699 of file `stl_set.h`.

```
4.688.4.36  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            const_iterator std::set<_Key, _Compare, _Alloc >::upper_bound ( const key_type & __x ) const  [inline]
```

Finds the end of a subsequence matching given key.

## Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

## Returns

Iterator pointing to the first element greater than key, or end().

Definition at line 703 of file `stl_set.h`.

```
4.688.4.37  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            value_compare std::set<_Key, _Compare, _Alloc >::value_comp ( ) const  [inline]
```

Returns the comparison object with which the set was constructed.

Definition at line 285 of file `stl_set.h`.

The documentation for this class was generated from the following file:

- [stl\\_set.h](#)

## 4.689 `std::shared_ptr<_Tp>` Class Template Reference

Inherits `std::__shared_ptr<_Tp, _Lp>`.

## Public Types

- typedef `_Tp element_type`

## Public Member Functions

- constexpr `shared_ptr()` noexcept
- `shared_ptr` (const `shared_ptr` &) noexcept=default
- template<typename `_Tp1` > `shared_ptr` (`_Tp1 *__p`)
- template<typename `_Tp1`, typename `_Deleter` > `shared_ptr` (`_Tp1 *__p`, `_Deleter __d`)
- template<typename `_Deleter` > `shared_ptr` (nullptr\_t `__p`, `_Deleter __d`)
- template<typename `_Tp1`, typename `_Deleter`, typename `_Alloc` > `shared_ptr` (`_Tp1 *__p`, `_Deleter __d`, `_Alloc __a`)
- template<typename `_Deleter`, typename `_Alloc` > `shared_ptr` (nullptr\_t `__p`, `_Deleter __d`, `_Alloc __a`)
- template<typename `_Tp1` > `shared_ptr` (const `shared_ptr`<`_Tp1`> &`__r`, `_Tp *__p`) noexcept

- `template<typename _Tp1, typename = typename std::enable_if<std::is_convertible<_Tp1*, _Tp*>::value>::type> shared_ptr (const shared_ptr< _Tp1 > &__r) noexcept`
- `shared_ptr (shared_ptr &&__r) noexcept`
- `template<typename _Tp1, typename = typename std::enable_if<std::is_convertible<_Tp1*, _Tp*>::value>::type> shared_ptr (shared_ptr< _Tp1 > &&__r) noexcept`
- `template<typename _Tp1 > shared_ptr (const weak_ptr< _Tp1 > &__r)`
- `template<typename _Tp1, typename _Del > shared_ptr (std::unique_ptr< _Tp1, _Del > &&__r)`
- `constexpr shared_ptr (nullptr_t __p) noexcept`
- `template<typename _Tp1 > shared_ptr (std::auto_ptr< _Tp1 > &&__r)`
- `_Tp * get () const noexcept`
- `operator bool () const`
- `std::add_lvalue_reference< _Tp >::type operator* () const noexcept`
- `_Tp * operator-> () const noexcept`
- `shared_ptr & operator= (const shared_ptr &) noexcept=default`
- `template<typename _Tp1 > shared_ptr & operator= (const shared_ptr< _Tp1 > &__r) noexcept`
- `shared_ptr & operator= (shared_ptr && __r) noexcept`
- `template<class _Tp1 > shared_ptr & operator= (shared_ptr< _Tp1 > &&__r) noexcept`
- `template<typename _Tp1, typename _Del > shared_ptr & operator= (std::unique_ptr< _Tp1, _Del > &&__r)`
- `template<typename _Tp1 > bool owner_before (__shared_ptr< _Tp1, _Lp > const &__rhs) const`
- `template<typename _Tp1 > bool owner_before (__weak_ptr< _Tp1, _Lp > const &__rhs) const`
- `void reset () noexcept`
- `template<typename _Tp1 > void reset (_Tp1 *__p)`
- `template<typename _Tp1, typename _Deleter > void reset (_Tp1 *__p, _Deleter __d)`
- `template<typename _Tp1, typename _Deleter, typename _Alloc > void reset (_Tp1 *__p, _Deleter __d, _Alloc __a)`
- `void swap (__shared_ptr< _Tp, _Lp > &__other) noexcept`
- `bool unique () const noexcept`
- `long use_count () const noexcept`

## Friends

- `template<typename _Tp1, typename _Alloc, typename... _Args> shared_ptr< _Tp1 > allocate_shared (const _Alloc &__a, _Args &&... __args)`

## 4.689.1 Detailed Description

`template<typename _Tp>class std::shared_ptr< _Tp >`

A smart pointer with reference-counted copy semantics.

The object pointed to is deleted when the last `shared_ptr` pointing to it is destroyed or reset.

Definition at line 93 of file `shared_ptr.h`.

## 4.689.2 Constructor & Destructor Documentation

### 4.689.2.1 `template<typename _Tp> constexpr std::shared_ptr< _Tp >::shared_ptr ( ) [inline], [noexcept]`

Construct an empty `shared_ptr`.

#### Postcondition

`use_count()==0 && get()==0`

Definition at line 100 of file `shared_ptr.h`.



4.689.2.2 `template<typename _Tp> template<typename _Tp1 > std::shared_ptr< _Tp >::shared_ptr ( _Tp1 * __p )`  
`[inline], [explicit]`

Construct a `shared_ptr` that owns the pointer `__p`.

## Parameters

<code>__p</code>	A pointer that is convertible to <code>element_type*</code> .
------------------	---

## Postcondition

`use_count() == 1 && get() == __p`

## Exceptions

<code>std::bad_alloc</code> , in	which case <code>delete __p</code> is called.
----------------------------------	---

Definition at line 112 of file `shared_ptr.h`.

4.689.2.3 `template<typename _Tp> template<typename _Tp1, typename _Deleter> std::shared_ptr< _Tp >::shared_ptr ( _Tp1 * __p, _Deleter __d ) [inline]`

Construct a `shared_ptr` that owns the pointer `__p` and the deleter `__d`.

## Parameters

<code>__p</code>	A pointer.
<code>__d</code>	A deleter.

## Postcondition

`use_count() == 1 && get() == __p`

## Exceptions

<code>std::bad_alloc</code> , in	which case <code>__d(__p)</code> is called.
----------------------------------	---

Requirements: `_Deleter`'s copy constructor and destructor must not throw

`__shared_ptr` will release `__p` by calling `__d(__p)`

Definition at line 129 of file `shared_ptr.h`.

4.689.2.4 `template<typename _Tp> template<typename _Deleter> std::shared_ptr< _Tp >::shared_ptr ( nullptr_t __p, _Deleter __d ) [inline]`

Construct a `shared_ptr` that owns a null pointer and the deleter `__d`.

## Parameters

<code>__p</code>	A null pointer constant.
<code>__d</code>	A deleter.

## Postcondition

`use_count() == 1 && get() == __p`

## Exceptions

<code>std::bad_alloc</code> , in	which case <code>__d(__p)</code> is called.
----------------------------------	---

Requirements: `_Deleter`'s copy constructor and destructor must not throw

The last owner will call `__d(__p)`

Definition at line 146 of file `shared_ptr.h`.

4.689.2.5 `template<typename _Tp> template<typename _Tp1 , typename _Deleter , typename _Alloc > std::shared_ptr<_Tp>::shared_ptr ( _Tp1 * __p, _Deleter __d, _Alloc __a ) [inline]`

Construct a `shared_ptr` that owns the pointer `__p` and the deleter `__d`.

## Parameters

<code>__p</code>	A pointer.
<code>__d</code>	A deleter.
<code>__a</code>	An allocator.

## Postcondition

`use_count() == 1 && get() == __p`

## Exceptions

<i>std::bad_alloc</i> , in	which case <code>__d(__p)</code> is called.
----------------------------	---

Requirements: `_Deleter`'s copy constructor and destructor must not throw `_Alloc`'s copy constructor and destructor must not throw.

`__shared_ptr` will release `__p` by calling `__d(__p)`

Definition at line 165 of file `shared_ptr.h`.

**4.689.2.6** `template<typename _Tp> template<typename _Deleter, typename _Alloc > std::shared_ptr< _Tp >::shared_ptr ( nullptr_t __p, _Deleter __d, _Alloc __a ) [inline]`

Construct a `shared_ptr` that owns a null pointer and the deleter `__d`.

## Parameters

<code>__p</code>	A null pointer constant.
<code>__d</code>	A deleter.
<code>__a</code>	An allocator.

## Postcondition

`use_count() == 1 && get() == __p`

## Exceptions

<i>std::bad_alloc</i> , in	which case <code>__d(__p)</code> is called.
----------------------------	---

Requirements: `_Deleter`'s copy constructor and destructor must not throw `_Alloc`'s copy constructor and destructor must not throw.

The last owner will call `__d(__p)`

Definition at line 184 of file `shared_ptr.h`.

**4.689.2.7** `template<typename _Tp> template<typename _Tp1 > std::shared_ptr< _Tp >::shared_ptr ( const shared_ptr< _Tp1 > & __r, _Tp* __p ) [inline], [noexcept]`

Constructs a `shared_ptr` instance that stores `__p` and shares ownership with `__r`.

## Parameters

<code>__r</code>	A <code>shared_ptr</code> .
<code>__p</code>	A pointer that will remain valid while <code>*__r</code> is valid.

**Postcondition**

```
get() == __p && use_count() == __r.use_count()
```

This can be used to construct a `shared_ptr` to a sub-object of an object managed by an existing `shared_ptr`.

```
shared_ptr< pair<int,int> > pii(new pair<int,int>());
shared_ptr<int> pi(pii, &pii->first);
assert(pii.use_count() == 2);
```

Definition at line 206 of file `shared_ptr.h`.

```
4.689.2.8 template<typename _Tp> template<typename _Tp1, typename = typename std::enable_if<std::is_convertible<_Tp1*,
    _Tp*>::value>::type> std::shared_ptr<_Tp>::shared_ptr( const shared_ptr<_Tp1> & __r )
    [inline], [noexcept]
```

If `__r` is empty, constructs an empty `shared_ptr`; otherwise construct a `shared_ptr` that shares ownership with `__r`.

**Parameters**

<code>__r</code>	A <code>shared_ptr</code> .
------------------	-----------------------------

**Postcondition**

```
get() == __r.get() && use_count() == __r.use_count()
```

Definition at line 218 of file `shared_ptr.h`.

```
4.689.2.9 template<typename _Tp> std::shared_ptr<_Tp>::shared_ptr( shared_ptr<_Tp> && __r ) [inline],
    [noexcept]
```

Move-constructs a `shared_ptr` instance from `__r`.

**Parameters**

<code>__r</code>	A <code>shared_ptr</code> rvalue.
------------------	-----------------------------------

**Postcondition**

\*this contains the old value of `__r`, `__r` is empty.

Definition at line 226 of file `shared_ptr.h`.

```
4.689.2.10 template<typename _Tp> template<typename _Tp1, typename = typename std::enable_if<std::is_convertible<_Tp1*,
    _Tp*>::value>::type> std::shared_ptr<_Tp>::shared_ptr( shared_ptr<_Tp1> && __r ) [inline],
    [noexcept]
```

Move-constructs a `shared_ptr` instance from `__r`.

**Parameters**

<code>__r</code>	A <code>shared_ptr</code> rvalue.
------------------	-----------------------------------

**Postcondition**

\*this contains the old value of `__r`, `__r` is empty.

Definition at line 236 of file `shared_ptr.h`.

```
4.689.2.11 template<typename _Tp> template<typename _Tp1> std::shared_ptr<_Tp>::shared_ptr( const weak_ptr<_Tp1> &__r ) [inline], [explicit]
```

Constructs a shared\_ptr that shares ownership with \_\_r and stores a copy of the pointer stored in \_\_r.

## Parameters

<code>__r</code>	A weak_ptr.
------------------	-------------

## Postcondition

`use_count() == __r.use_count()`

## Exceptions

<i>bad_weak_ptr</i>	when <code>__r.expired()</code> , in which case the constructor has no effect.
---------------------	--

Definition at line 248 of file `shared_ptr.h`.

4.689.2.12 `template<typename _Tp> constexpr std::shared_ptr<_Tp>::shared_ptr ( nullptr_t __p ) [inline],  
[noexcept]`

Construct an empty `shared_ptr`.

## Parameters

<code>__p</code>	A null pointer constant.
------------------	--------------------------

## Postcondition

`use_count() == 0 && get() == nullptr`

Definition at line 265 of file `shared_ptr.h`.

## 4.689.3 Friends And Related Function Documentation

4.689.3.1 `template<typename _Tp> template<typename _Tp1, typename _Alloc, typename... _Args> shared_ptr<_Tp1>  
allocate_shared ( const _Alloc & __a, _Args &&... __args ) [friend]`

Create an object that is owned by a `shared_ptr`.

## Parameters

<code>__a</code>	An allocator.
<code>__args</code>	Arguments for the <code>_Tp</code> object's constructor.

## Returns

A `shared_ptr` that owns the newly created object.

## Exceptions

<i>An</i>	exception thrown from <code>_Alloc::allocate</code> or from the constructor of <code>_Tp</code> .
-----------	---

A copy of `__a` will be used to allocate memory for the `shared_ptr` and the new object.

Definition at line 595 of file `shared_ptr.h`.

The documentation for this class was generated from the following files:

- [shared\\_ptr.h](#)
- [auto\\_ptr.h](#)

4.690 `std::shuffle_order_engine<_RandomNumberEngine, __k>` Class Template Reference

## Public Types

- `typedef _RandomNumberEngine::result_type` [result\\_type](#)

## Public Member Functions

- [shuffle\\_order\\_engine](#) ()
- [shuffle\\_order\\_engine](#) (const `_RandomNumberEngine` &\_\_rng)
- [shuffle\\_order\\_engine](#) (`_RandomNumberEngine` &&\_\_rng)
- [shuffle\\_order\\_engine](#) ([result\\_type](#) \_\_s)
- `template<typename _Sseq, typename = typename std::enable_if<!std::is_same<_Sseq, shuffle_order_engine>::value && !std::is_↵`  
same<\_Sseq, `_RandomNumberEngine`>::value>::type> [shuffle\\_order\\_engine](#) (`_Sseq` &\_\_q)
- const `_RandomNumberEngine` & [base](#) () const noexcept
- void [discard](#) (unsigned long long \_\_z)
- [result\\_type](#) [operator](#)() ()
- void [seed](#) ()
- void [seed](#) ([result\\_type](#) \_\_s)
- `template<typename _Sseq> void` [seed](#) (`_Sseq` &\_\_q)

## Static Public Member Functions

- static constexpr [result\\_type](#) [max](#) ()
- static constexpr [result\\_type](#) [min](#) ()

## Static Public Attributes

- static constexpr `size_t` [table\\_size](#)

## Friends

- `template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits> std::basic_ostream<_Char↵`  
T, `_Traits` > & [operator<<](#) (std::basic\_ostream<\_CharT, `_Traits` > &\_\_os, const [std::shuffle\\_order\\_engine<\\_RandomNumberEngine1, \\_\\_k1>](#) &\_\_x)
- bool [operator==](#) (const [shuffle\\_order\\_engine](#) &\_\_lhs, const [shuffle\\_order\\_engine](#) &\_\_rhs)
- `template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits> std::basic_istream<_CharT, ↵`  
`_Traits` > & [operator>>](#) (std::basic\_istream<\_CharT, `_Traits` > &\_\_is, [std::shuffle\\_order\\_engine<\\_Random↵](#)  
NumberEngine1, `__k1` > &\_\_x)

## 4.690.1 Detailed Description

`template<typename _RandomNumberEngine, size_t __k>class std::shuffle_order_engine<_RandomNumberEngine, __k>`

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits `__w`.

Definition at line 1292 of file random.h.



#### 4.690.2 Member Typedef Documentation

4.690.2.1 `template<typename _RandomNumberEngine, size_t __k> typedef _RandomNumberEngine::result_type  
std::shuffle_order_engine< _RandomNumberEngine, __k >::result_type`

The type of the generated random value.

Definition at line 1295 of file random.h.

#### 4.690.3 Constructor & Destructor Documentation

4.690.3.1 `template<typename _RandomNumberEngine, size_t __k> std::shuffle_order_engine< _RandomNumberEngine,  
__k >::shuffle_order_engine ( ) [inline]`

Constructs a default shuffle\_order\_engine engine.

The underlying engine is default constructed as well.

Definition at line 1308 of file random.h.

4.690.3.2 `template<typename _RandomNumberEngine, size_t __k> std::shuffle_order_engine< _RandomNumberEngine,  
__k >::shuffle_order_engine ( const _RandomNumberEngine & __rng ) [inline],[explicit]`

Copy constructs a shuffle\_order\_engine engine.

Copies an existing base class random number generator.

##### Parameters

<code>__rng</code>	An existing (base class) engine object.
--------------------	---

Definition at line 1319 of file random.h.

4.690.3.3 `template<typename _RandomNumberEngine, size_t __k> std::shuffle_order_engine< _RandomNumberEngine,  
__k >::shuffle_order_engine ( _RandomNumberEngine && __rng ) [inline],[explicit]`

Move constructs a shuffle\_order\_engine engine.

Copies an existing base class random number generator.

##### Parameters

<code>__rng</code>	An existing (base class) engine object.
--------------------	---

Definition at line 1330 of file random.h.

4.690.3.4 `template<typename _RandomNumberEngine, size_t __k> std::shuffle_order_engine< _RandomNumberEngine,  
__k >::shuffle_order_engine ( result_type __s ) [inline],[explicit]`

Seed constructs a shuffle\_order\_engine engine.

Constructs the underlying generator engine seeded with `__s`.

##### Parameters

<code>__s</code>	A seed value for the base class engine.
------------------	---

Definition at line 1341 of file random.h.

```
4.690.3.5 template<typename _RandomNumberEngine, size_t __k> template<typename _Sseq, typename =  
    typename std::enable_if<!std::is_same<_Sseq, shuffle_order_engine>::value && !std::is_same<_Sseq,  
    _RandomNumberEngine>::value> ::type> std::shuffle_order_engine<_RandomNumberEngine, __k  
    >::shuffle_order_engine ( _Sseq & __q ) [inline], [explicit]
```

Generator construct a shuffle\_order\_engine engine.

## Parameters

<code>__q</code>	A seed sequence.
------------------	------------------

Definition at line 1355 of file random.h.

## 4.690.4 Member Function Documentation

4.690.4.1 `template<typename _RandomNumberEngine, size_t __k> const _RandomNumberEngine& std::shuffle_order_engine< _RandomNumberEngine, __k >::base ( ) const [inline], [noexcept]`

Gets a const reference to the underlying generator engine object.

Definition at line 1398 of file random.h.

4.690.4.2 `template<typename _RandomNumberEngine, size_t __k> void std::shuffle_order_engine< _RandomNumberEngine, __k >::discard ( unsigned long long __z ) [inline]`

Discard a sequence of random numbers.

Definition at line 1419 of file random.h.

4.690.4.3 `template<typename _RandomNumberEngine, size_t __k> static constexpr result_type std::shuffle_order_engine< _RandomNumberEngine, __k >::max ( ) [inline], [static]`

Gets the maximum value in the generated random number range.

Definition at line 1412 of file random.h.

References `std::max()`.

4.690.4.4 `template<typename _RandomNumberEngine, size_t __k> static constexpr result_type std::shuffle_order_engine< _RandomNumberEngine, __k >::min ( ) [inline], [static]`

Gets the minimum value in the generated random number range.

Definition at line 1405 of file random.h.

References `std::min()`.

4.690.4.5 `template<typename _RandomNumberEngine, size_t __k> result_type std::shuffle_order_engine< _RandomNumberEngine, __k >::operator() ( )`

Gets the next value in the generated random number sequence.

4.690.4.6 `template<typename _RandomNumberEngine, size_t __k> void std::shuffle_order_engine< _RandomNumberEngine, __k >::seed ( ) [inline]`

Reseeds the `shuffle_order_engine` object with the default seed for the underlying base class generator engine.

Definition at line 1364 of file random.h.

4.690.4.7 `template<typename _RandomNumberEngine, size_t __k> void std::shuffle_order_engine< _RandomNumberEngine, __k >::seed ( result_type __s ) [inline]`

Reseeds the `shuffle_order_engine` object with the default seed for the underlying base class generator engine.

Definition at line 1375 of file random.h.

```
4.690.4.8  template<typename _RandomNumberEngine, size_t __k> template<typename _Sseq > void  
           std::shuffle_order_engine<_RandomNumberEngine, __k>::seed ( _Sseq & __q ) [inline]
```

Reseeds the `shuffle_order_engine` object with the given seed sequence.

## Parameters

<code>__q</code>	A seed generator function.
------------------	----------------------------

Definition at line 1388 of file random.h.

## 4.690.5 Friends And Related Function Documentation

4.690.5.1 `template<typename _RandomNumberEngine, size_t __k> template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits> std::basic_ostream<_CharT, _Traits>& operator<< ( std::basic_ostream<_CharT, _Traits> & __os, const std::shuffle_order_engine<_RandomNumberEngine1, __k1> & __x )`  
`[friend]`

Inserts the current state of a shuffle\_order\_engine random number generator engine `__x` into the output stream `__os`.

## Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A shuffle_order_engine random number generator engine.

## Returns

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

4.690.5.2 `template<typename _RandomNumberEngine, size_t __k> bool operator== ( const shuffle_order_engine<_RandomNumberEngine, __k> & __lhs, const shuffle_order_engine<_RandomNumberEngine, __k> & __rhs )`  
`[friend]`

Compares two shuffle\_order\_engine random number generator objects of the same type for equality.

## Parameters

<code>__lhs</code>	A shuffle_order_engine random number generator object.
<code>__rhs</code>	Another shuffle_order_engine random number generator object.

## Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 1443 of file random.h.

4.690.5.3 `template<typename _RandomNumberEngine, size_t __k> template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<_CharT, _Traits> & __is, std::shuffle_order_engine<_RandomNumberEngine1, __k1> & __x )` `[friend]`

Extracts the current state of a % subtract\_with\_carry\_engine random number generator engine `__x` from the input stream `__is`.

## Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A shuffle_order_engine random number generator engine.

## Returns

The input stream with the state of `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

## 4.691 std::slice Class Reference

## Public Member Functions

- [slice](#) ()
- [slice](#) (size\_t \_\_o, size\_t \_\_d, size\_t \_\_s)
- size\_t [size](#) () const
- size\_t [start](#) () const
- size\_t [stride](#) () const

## 4.691.1 Detailed Description

Class defining one-dimensional subset of an array.

The slice class represents a one-dimensional subset of an array, specified by three parameters: start offset, size, and stride. The start offset is the index of the first element of the array that is part of the subset. The size is the total number of elements in the subset. Stride is the distance between each successive array element to include in the subset.

For example, with an array of size 10, and a slice with offset 1, size 3 and stride 2, the subset consists of array elements 1, 3, and 5.

Definition at line 59 of file `slice_array.h`.

The documentation for this class was generated from the following file:

- [slice\\_array.h](#)

## 4.692 std::slice\_array&lt; \_Tp &gt; Class Template Reference

## Public Types

- typedef `_Tp` **value\_type**

## Public Member Functions

- [slice\\_array](#) (const [slice\\_array](#) &)
- void [operator%=>](#) (const valarray< \_Tp > &) const
- template<class \_Dom > void [operator%=>](#) (const \_Expr< \_Dom, \_Tp > &) const
- void [operator&=>](#) (const valarray< \_Tp > &) const
- template<class \_Dom > void [operator&=>](#) (const \_Expr< \_Dom, \_Tp > &) const
- void [operator\\*=>](#) (const valarray< \_Tp > &) const
- template<class \_Dom > void [operator\\*=>](#) (const \_Expr< \_Dom, \_Tp > &) const
- void [operator+=>](#) (const valarray< \_Tp > &) const
- template<class \_Dom > void [operator+=>](#) (const \_Expr< \_Dom, \_Tp > &) const

- void [operator=](#) (const valarray< \_Tp > &) const
- template<class \_Dom > void **operator=** (const \_Expr< \_Dom, \_Tp > &) const
- void [operator/=](#) (const valarray< \_Tp > &) const
- template<class \_Dom > void **operator/=** (const \_Expr< \_Dom, \_Tp > &) const
- void [operator<<=](#) (const valarray< \_Tp > &) const
- template<class \_Dom > void **operator<<=** (const \_Expr< \_Dom, \_Tp > &) const
- [slice\\_array](#) & [operator=](#) (const [slice\\_array](#) &)
- void [operator=](#) (const valarray< \_Tp > &) const
- void [operator=](#) (const \_Tp &) const
- template<class \_Dom > void **operator=** (const \_Expr< \_Dom, \_Tp > &) const
- void [operator>>=](#) (const valarray< \_Tp > &) const
- template<class \_Dom > void **operator>>=** (const \_Expr< \_Dom, \_Tp > &) const
- void [operator^=](#) (const valarray< \_Tp > &) const
- template<class \_Dom > void **operator^=** (const \_Expr< \_Dom, \_Tp > &) const
- void [operator|=](#) (const valarray< \_Tp > &) const
- template<class \_Dom > void **operator|=** (const \_Expr< \_Dom, \_Tp > &) const

#### Friends

- class **valarray**< \_Tp >

#### 4.692.1 Detailed Description

template<typename \_Tp>class std::slice\_array< \_Tp >

Reference to one-dimensional subset of an array.

A slice\_array is a reference to the actual elements of an array specified by a slice. The way to get a slice\_array is to call operator[](slice) on a valarray. The returned slice\_array then permits carrying operations out on the referenced subset of elements in the original valarray. For example, operator+=(valarray) will add values to the subset of elements in the underlying valarray this slice\_array refers to.

#### Parameters

<i>Tp</i>	Element type.
-----------	---------------

Definition at line 123 of file slice\_array.h.

The documentation for this class was generated from the following file:

- [slice\\_array.h](#)

#### 4.693 std::stack< \_Tp, \_Sequence > Class Template Reference

##### Public Types

- typedef \_Sequence::const\_reference **const\_reference**
- typedef \_Sequence **container\_type**
- typedef \_Sequence::reference **reference**
- typedef \_Sequence::size\_type **size\_type**
- typedef \_Sequence::value\_type **value\_type**

## Public Member Functions

- `stack` (`const _Sequence &__c`)
- `stack` (`_Sequence &&__c=_Sequence()`)
- `template<typename... _Args> void emplace` (`_Args &&... __args`)
- `bool empty` () `const`
- `void pop` ()
- `void push` (`const value_type &__x`)
- `void push` (`value_type &&__x`)
- `size_type size` () `const`
- `void swap` (`stack &__s`) `noexcept(noexcept(swap(c, __s.c)))`
- `reference top` ()
- `const_reference top` () `const`

## Protected Attributes

- `_Sequence c`

## Friends

- `template<typename _Tp1, typename _Seq1> bool operator<` (`const stack<_Tp1, _Seq1> &`, `const stack<_Tp1, _Seq1> &`)
- `template<typename _Tp1, typename _Seq1> bool operator==` (`const stack<_Tp1, _Seq1> &`, `const stack<_Tp1, _Seq1> &`)

## 4.693.1 Detailed Description

`template<typename _Tp, typename _Sequence = deque<_Tp>>class std::stack<_Tp, _Sequence>`

A standard container giving FILO behavior.

## Template Parameters

<code>_Tp</code>	Type of element.
<code>_Sequence</code>	Type of underlying sequence, defaults to <code>deque&lt;_Tp&gt;</code> .

Meets many of the requirements of a `container`, but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-last-out stack behavior.

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::deque`, but it can be any type that supports `back`, `push_back`, and `pop_front`, such as `std::list`, `std::vector`, or an appropriate user-defined type.

Members not found in *normal* containers are `container_type`, which is a typedef for the second `Sequence` parameter, and `push`, `pop`, and `top`, which are standard stack/FILO operations.

Definition at line 96 of file `stl_stack.h`.



#### 4.693.2 Constructor & Destructor Documentation

4.693.2.1 `template<typename _Tp, typename _Sequence = deque<_Tp>> std::stack<_Tp, _Sequence>::stack ( const _Sequence &__c ) [inline], [explicit]`

Default constructor creates no elements.

Definition at line 134 of file `stl_stack.h`.

#### 4.693.3 Member Function Documentation

4.693.3.1 `template<typename _Tp, typename _Sequence = deque<_Tp>> bool std::stack<_Tp, _Sequence>::empty ( ) const [inline]`

Returns true if the stack is empty.

Definition at line 146 of file `stl_stack.h`.

4.693.3.2 `template<typename _Tp, typename _Sequence = deque<_Tp>> void std::stack<_Tp, _Sequence>::pop ( ) [inline]`

Removes first element.

This is a typical stack operation. It shrinks the stack by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop()` is called.

Definition at line 212 of file `stl_stack.h`.

4.693.3.3 `template<typename _Tp, typename _Sequence = deque<_Tp>> void std::stack<_Tp, _Sequence>::push ( const value_type &__x ) [inline]`

Add data to the top of the stack.

##### Parameters

<code>__x</code>	Data to be added.
------------------	-------------------

This is a typical stack operation. The function creates an element at the top of the stack and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.

Definition at line 186 of file `stl_stack.h`.

4.693.3.4 `template<typename _Tp, typename _Sequence = deque<_Tp>> size_type std::stack<_Tp, _Sequence>::size ( ) const [inline]`

Returns the number of elements in the stack.

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

4.693.3.5 `template<typename _Tp, typename _Sequence = deque<_Tp>> reference std::stack<_Tp, _Sequence>::top ( ) [inline]`

Returns a read/write reference to the data at the first element of the stack.

Definition at line 159 of file `stl_stack.h`.

4.693.3.6 `template<typename _Tp, typename _Sequence = deque<_Tp>> const_reference std::stack<_Tp, _Sequence>::top  
( ) const [inline]`

Returns a read-only (constant) reference to the data at the first element of the stack.

Definition at line 170 of file `stl_stack.h`.

The documentation for this class was generated from the following file:

- [stl\\_stack.h](#)

## 4.694 `std::student_t_distribution<_RealType>` Class Template Reference

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

### Public Member Functions

- **`student_t_distribution`** (`_RealType __n=_RealType(1)`)
- **`student_t_distribution`** (`const` [param\\_type](#) &`__p`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator> void __generate` (`_ForwardIterator __f,`  
`_ForwardIterator __t, _UniformRandomNumberGenerator &__urng`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator> void __generate` (`_ForwardIterator __f,`  
`_ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const` [param\\_type](#) &`__p`)
- `template<typename _UniformRandomNumberGenerator> void __generate` ([result\\_type](#) \*`__f, result_type *__t, _UniformRandomNumberGenerator &__urng)`
- `template<typename _UniformRandomNumberGenerator> void __generate` ([result\\_type](#) \*`__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param\_type &__p)`
- [result\\_type](#) `max` () `const`
- [result\\_type](#) `min` () `const`
- `_RealType` `n` () `const`
- `template<typename _UniformRandomNumberGenerator> result_type operator()` (`_UniformRandomNumberGenerator &__urng`)
- `template<typename _UniformRandomNumberGenerator> result_type operator()` (`_UniformRandomNumberGenerator &__urng, const` [param\\_type](#) &`__p`)
- [param\\_type](#) `param` () `const`
- `void` [param](#) (`const` [param\\_type](#) &`__param`)
- `void` [reset](#) ()

### Friends

- `template<typename _RealType1, typename _CharT, typename _Traits> std::basic_ostream<_CharT, _Traits> &`  
`operator<<` (`std::basic_ostream<_CharT, _Traits> &__os, const` `std::student_t_distribution<_RealType1>`  
`> &__x`)
- `bool` `operator==` (`const` `student_t_distribution` &`__d1, const` `student_t_distribution` &`__d2`)
- `template<typename _RealType1, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits> &`  
`operator>>` (`std::basic_istream<_CharT, _Traits> &__is, std::student_t_distribution<_RealType1>`  
`> &__x`)

## 4.694.1 Detailed Description

```
template<typename _RealType = double>class std::student_t_distribution< _RealType >
```

A student\_t\_distribution random number distribution.

The formula for the normal probability mass function is:

$$p(x|n) = \frac{1}{\sqrt{(n\pi)}} \frac{\Gamma((n+1)/2)}{\Gamma(n/2)} \left(1 + \frac{x^2}{n}\right)^{-(n+1)/2}$$

Definition at line 3354 of file random.h.

## 4.694.2 Member Typedef Documentation

```
4.694.2.1 template<typename _RealType = double> typedef _RealType std::student_t_distribution< _RealType
>::result_type
```

The type of the range of the distribution.

Definition at line 3357 of file random.h.

## 4.694.3 Member Function Documentation

```
4.694.3.1 template<typename _RealType = double> result_type std::student_t_distribution< _RealType >::max ( ) const
[inline]
```

Returns the least upper bound value of the distribution.

Definition at line 3437 of file random.h.

References std::max().

```
4.694.3.2 template<typename _RealType = double> result_type std::student_t_distribution< _RealType >::min ( ) const
[inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 3430 of file random.h.

```
4.694.3.3 template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type
std::student_t_distribution< _RealType >::operator() ( _UniformRandomNumberGenerator & __urng )
[inline]
```

Generating functions.

Definition at line 3445 of file random.h.

```
4.694.3.4 template<typename _RealType = double> param_type std::student_t_distribution< _RealType >::param ( )
const [inline]
```

Returns the parameter set of the distribution.

Definition at line 3415 of file random.h.

```
4.694.3.5  template<typename _RealType = double> void std::student_t_distribution<_RealType>::param ( const  
           param_type & __param ) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 3423 of file random.h.

4.694.3.6 `template<typename _RealType = double> void std::student_t_distribution< _RealType >::reset ( )`  
`[inline]`

Resets the distribution state.

Definition at line 3398 of file random.h.

References `std::normal_distribution< _RealType >::reset()`, and `std::gamma_distribution< _RealType >::reset()`.

**4.694.4 Friends And Related Function Documentation**

4.694.4.1 `template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > std::basic_ostream<_CharT, _Traits>& operator<< ( std::basic_ostream<_CharT, _Traits > & __os, const std::student_t_distribution< _RealType1 > & __x )` `[friend]`

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

**Parameters**

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

**Returns**

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

4.694.4.2 `template<typename _RealType = double> bool operator==( const student_t_distribution< _RealType > & __d1, const student_t_distribution< _RealType > & __d2 )` `[friend]`

Return true if two Student t distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3494 of file random.h.

4.694.4.3 `template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<_CharT, _Traits > & __is, std::student_t_distribution< _RealType1 > & __x )` `[friend]`

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

**Parameters**

<code>__is</code>	An input stream.
<code>__x</code>	A <code>student_t_distribution</code> random number generator engine.

**Returns**

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

4.695 `std::student_t_distribution<_RealType>::param_type` Struct Reference

## Public Types

- typedef `student_t_distribution<_RealType>` **distribution\_type**

## Public Member Functions

- **param\_type** (`_RealType __n=_RealType(1)`)
- `_RealType n` () const

## Friends

- bool **operator==** (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

## 4.695.1 Detailed Description

template<typename `_RealType` = double>struct `std::student_t_distribution<_RealType>::param_type`

Parameter type.

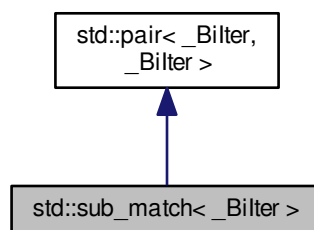
Definition at line 3363 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.696 `std::sub_match<_Bilter>` Class Template Reference

Inheritance diagram for `std::sub_match<_Bilter>`:



## Public Types

- typedef `__iter_traits::difference_type` **difference\_type**
- typedef `_Bilter` **first\_type**

- typedef `_Bilter` **iterator**
- typedef `_Bilter` `second_type`
- typedef `std::basic_string`< `value_type` > **string\_type**
- typedef `__iter_traits::value_type` **value\_type**

#### Public Member Functions

- int `compare` (const `sub_match` &\_\_s) const
- int `compare` (const `string_type` &\_\_s) const
- int `compare` (const `value_type` \*\_\_s) const
- `difference_type` `length` () const
- `operator string_type` () const
- `string_type` `str` () const
- void **swap** (`pair` &\_\_p) noexcept(noexcept(swap(`first`, \_\_p.first))&&noexcept(swap(`second`, \_\_p.second)))

#### Public Attributes

- `_Bilter` `first`
- bool **matched**
- `_Bilter` `second`

#### 4.696.1 Detailed Description

`template<typename _Bilter> class std::sub_match< _Bilter >`

A sequence of characters matched by a particular marked sub-expression.

An object of this class is essentially a pair of iterators marking a matched subexpression within a regular expression pattern match. Such objects can be converted to and compared with `std::basic_string` objects of a similar base character type as the pattern matched by the regular expression.

The iterators that make up the pair are the usual half-open interval referencing the actual original pattern matched.

Definition at line 741 of file `regex.h`.

#### 4.696.2 Member Typedef Documentation

4.696.2.1 `typedef _Bilter std::pair< _Bilter , _Bilter >::second_type` `[inherited]`

`first_type` is the first bound type

Definition at line 99 of file `stl_pair.h`.

#### 4.696.3 Member Function Documentation

4.696.3.1 `template<typename _Bilter> int std::sub_match< _Bilter >::compare ( const sub_match< _Bilter > & __s )`  
`const` `[inline]`

Compares this and another matched sequence.

## Parameters

<code>__s</code>	Another matched sequence to compare to this one.
------------------	--

## Return values

<code>&lt;0</code>	this matched sequence will collate before <code>__s</code> .
<code>=0</code>	this matched sequence is equivalent to <code>__s</code> .
<code>&lt;0</code>	this matched sequence will collate after <code>__s</code> .

Definition at line 802 of file regex.h.

Referenced by `std::operator!=()`, `std::operator<()`, `std::operator<=()`, `std::operator==()`, `std::operator>()`, and `std::operator>=()`.

4.696.3.2 `template<typename _Bilter> int std::sub_match<_Bilter>::compare ( const string_type & __s ) const`  
`[inline]`

Compares this `sub_match` to a string.

## Parameters

<code>__s</code>	A string to compare to this <code>sub_match</code> .
------------------	--

## Return values

<code>&lt;0</code>	this matched sequence will collate before <code>__s</code> .
<code>=0</code>	this matched sequence is equivalent to <code>__s</code> .
<code>&lt;0</code>	this matched sequence will collate after <code>__s</code> .

Definition at line 815 of file regex.h.

4.696.3.3 `template<typename _Bilter> int std::sub_match<_Bilter>::compare ( const value_type * __s ) const`  
`[inline]`

Compares this `sub_match` to a C-style string.

## Parameters

<code>__s</code>	A C-style string to compare to this <code>sub_match</code> .
------------------	--

## Return values

<code>&lt;0</code>	this matched sequence will collate before <code>__s</code> .
<code>=0</code>	this matched sequence is equivalent to <code>__s</code> .
<code>&lt;0</code>	this matched sequence will collate after <code>__s</code> .

Definition at line 828 of file regex.h.

4.696.3.4 `template<typename _Bilter> difference_type std::sub_match<_Bilter>::length ( ) const` `[inline]`

Gets the length of the matching sequence.

Definition at line 759 of file regex.h.

4.696.3.5 `template<typename _Bilter> std::sub_match<_Bilter>::operator string_type ( ) const` `[inline]`

Gets the matching sequence as a string.



**Returns**

the matching sequence as a string.

This is the implicit conversion operator. It is identical to the `str()` member function except that it will want to pop up in unexpected places and cause a great deal of confusion and cursing from the unwary.

Definition at line 772 of file `regex.h`.

**4.696.3.6** `template<typename _Bilter> string_type std::sub_match<_Bilter>::str ( ) const [inline]`

Gets the matching sequence as a string.

**Returns**

the matching sequence as a string.

Definition at line 785 of file `regex.h`.

Referenced by `std::sub_match<_Bi_iter>::compare()`, and `std::operator<<()`.

**4.696.4 Member Data Documentation**

**4.696.4.1** `_Bilter std::pair<_Bilter, _Bilter>::first [inherited]`

`second_type` is the second bound type

Definition at line 101 of file `stl_pair.h`.

Referenced by `std::sub_match<_Bi_iter>::length()`, `std::sub_match<_Bi_iter>::operator string_type()`, and `std::sub_match<_Bi_iter>::str()`.

**4.696.4.2** `_Bilter std::pair<_Bilter, _Bilter>::second [inherited]`

`first` is a copy of the first object

Definition at line 102 of file `stl_pair.h`.

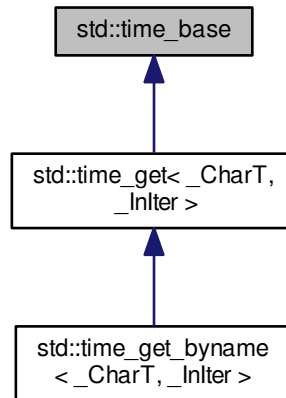
Referenced by `std::sub_match<_Bi_iter>::length()`, `std::sub_match<_Bi_iter>::operator string_type()`, and `std::sub_match<_Bi_iter>::str()`.

The documentation for this class was generated from the following file:

- [regex.h](#)

## 4.697 std::time\_base Class Reference

Inheritance diagram for std::time\_base:



## Public Types

- enum **dateorder** { **no\_order**, **dmy**, **mdy**, **ymd**, **ydm** }

## 4.697.1 Detailed Description

Time format ordering data.

This class provides an enum representing different orderings of time: day, month, and year.

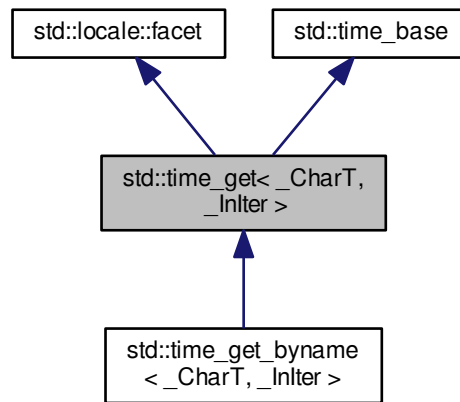
Definition at line 52 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

#### 4.698 `std::time_get<_CharT, _InIter>` Class Template Reference

Inheritance diagram for `std::time_get<_CharT, _InIter>`:



##### Public Types

- typedef `basic_string<_CharT>` `__string_type`
- enum `dateorder` { `no_order`, `dmy`, `mdy`, `ymd`, `ydm` }
- typedef `_CharT` `char_type`
- typedef `_InIter` `iter_type`

##### Public Member Functions

- `time_get` (`size_t` \_\_refs=0)
- `dateorder` `date_order` () const
- `iter_type` `get_date` (`iter_type` \_\_beg, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, `tm` \* \_\_tm) const
- `iter_type` `get_monthname` (`iter_type` \_\_beg, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, `tm` \* \_\_tm) const
- `iter_type` `get_time` (`iter_type` \_\_beg, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, `tm` \* \_\_tm) const
- `iter_type` `get_weekday` (`iter_type` \_\_beg, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, `tm` \* \_\_tm) const
- `iter_type` `get_year` (`iter_type` \_\_beg, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, `tm` \* \_\_tm) const

##### Static Public Attributes

- static `locale::id` `id`

## Protected Member Functions

- virtual `~time_get()`
- `iter_type M_extract_name(iter_type __beg, iter_type __end, int &__member, const _CharT **__names, size_t __indexlen, ios_base &__io, ios_base::iostate &__err) const`
- `iter_type M_extract_num(iter_type __beg, iter_type __end, int &__member, int __min, int __max, size_t __len, ios_base &__io, ios_base::iostate &__err) const`
- `iter_type M_extract_via_format(iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm, const _CharT *__format) const`
- `iter_type M_extract_wday_or_month(iter_type __beg, iter_type __end, int &__member, const _CharT **__names, size_t __indexlen, ios_base &__io, ios_base::iostate &__err) const`
- virtual `dateorder do_date_order() const`
- virtual `iter_type do_get_date(iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`
- virtual `iter_type do_get_monthname(iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`
- virtual `iter_type do_get_time(iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`
- virtual `iter_type do_get_weekday(iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`
- virtual `iter_type do_get_year(iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`

## Static Protected Member Functions

- static `_c_locale _S_clone_c_locale(_c_locale &__cloc) throw()`
- static void `_S_create_c_locale(_c_locale &__cloc, const char *__s, _c_locale __old=0)`
- static void `_S_destroy_c_locale(_c_locale &__cloc)`
- static `_c_locale _S_get_c_locale()`
- static const char \* `_S_get_c_name() throw()`
- static `_c_locale _S_lc_ctype_c_locale(_c_locale __cloc, const char *__s)`

## 4.698.1 Detailed Description

template<typename \_CharT, typename \_InIter> class std::time\_get<\_CharT, \_InIter>

Primary class template time\_get.

This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.

The time\_get template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the time\_get facet.

Definition at line 368 of file locale\_facets\_nonio.h.

## 4.698.2 Member Typedef Documentation

4.698.2.1 template<typename \_CharT, typename \_InIter> typedef \_CharT std::time\_get<\_CharT, \_InIter>::char\_type

Public typedefs.

Definition at line 374 of file locale\_facets\_nonio.h.

4.698.2.2 `template<typename _CharT, typename _Inlter > typedef _Inlter std::time_get< _CharT, _Inlter >::iter_type`

Public typedefs.

Definition at line 375 of file `locale_facets_nonio.h`.

#### 4.698.3 Constructor & Destructor Documentation

4.698.3.1 `template<typename _CharT, typename _Inlter > std::time_get< _CharT, _Inlter >::time_get ( size_t __refs = 0 )`  
`[inline], [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

##### Parameters

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

Definition at line 390 of file `locale_facets_nonio.h`.

4.698.3.2 `template<typename _CharT, typename _Inlter > virtual std::time_get< _CharT, _Inlter >::~time_get ( )`  
`[inline], [protected], [virtual]`

Destructor.

Definition at line 546 of file `locale_facets_nonio.h`.

#### 4.698.4 Member Function Documentation

4.698.4.1 `template<typename _CharT, typename _Inlter > dateorder std::time_get< _CharT, _Inlter >::date_order ( ) const`  
`[inline]`

Return preferred order of month, day, and year.

This function returns an enum from `timebase::dateorder` giving the preferred ordering if the format `x` given to `time_put::put()` only uses month, day, and year. If the format `x` for the associated locale uses other fields, this function returns `timebase::dateorder::noorder`.

NOTE: The library always returns `noorder` at the moment.

##### Returns

A member of `timebase::dateorder`.

Definition at line 407 of file `locale_facets_nonio.h`.

References `std::time_get< _CharT, _Inlter >::do_date_order()`.

4.698.4.2 `template<typename _CharT, typename _Inlter > virtual dateorder std::time_get< _CharT, _Inlter >::do_date_order ( ) const`  
`[protected], [virtual]`

Return preferred order of month, day, and year.

This function returns an enum from `timebase::dateorder` giving the preferred ordering if the format `x` given to `time_put::put()` only uses month, day, and year. This function is a hook for derived classes to change the value returned.

**Returns**

A member of timebase::dateorder.

Referenced by std::time\_get< \_CharT, \_InIter >::date\_order().

4.698.4.3 `template<typename _CharT, typename _InIter > virtual iter_type std::time_get< _CharT, _InIter >::do_get_date  
( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const  
[protected], [virtual]`

Parse input date string.

This function parses a date according to the format *X* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

**See also**

get\_date() for details.

**Parameters**

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

**Returns**

Iterator to first char beyond date string.

Referenced by std::time\_get< \_CharT, \_InIter >::get\_date().

4.698.4.4 `template<typename _CharT, typename _InIter > virtual iter_type std::time_get< _CharT, _InIter  
>::do_get_monthname( iter_type __beg, iter_type __end, ios_base & , ios_base::iostate & __err, tm * __tm )  
const [protected], [virtual]`

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

**See also**

get\_monthname() for details.

**Parameters**

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.

<code>__tm</code>	Pointer to struct tm to fill in.
-------------------	----------------------------------

**Returns**

Iterator to first char beyond month name.

Referenced by `std::time_get<_CharT, _InIter>::get_monthname()`.

```
4.698.4.5 template<typename _CharT, typename _InIter> virtual iter_type std::time_get<_CharT, _InIter>::do_get_time
( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const
[protected], [virtual]
```

Parse input time string.

This function parses a time according to the format `x` and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

**See also**

`get_time()` for details.

**Parameters**

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

**Returns**

Iterator to first char beyond time string.

Referenced by `std::time_get<_CharT, _InIter>::get_time()`.

```
4.698.4.6 template<typename _CharT, typename _InIter> virtual iter_type std::time_get<_CharT, _InIter>::do_get_weekday
( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const
[protected], [virtual]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

**See also**

`get_weekday()` for details.

**Parameters**

<code>__beg</code>	Start of string to parse.
--------------------	---------------------------

<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

**Returns**

Iterator to first char beyond weekday name.

Referenced by `std::time_get< _CharT, _InIter >::get_weekday()`.

```
4.698.4.7 template<typename _CharT, typename _InIter> virtual iter_type std::time_get< _CharT, _InIter >::do_get_year
( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const
[protected], [virtual]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

**See also**

`get_year()` for details.

**Parameters**

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

**Returns**

Iterator to first char beyond year.

Referenced by `std::time_get< _CharT, _InIter >::get_year()`.

```
4.698.4.8 template<typename _CharT, typename _InIter> iter_type std::time_get< _CharT, _InIter >::get_date ( iter_type
__beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const [inline]
```

Parse input date string.

This function parses a date according to the format *x* and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_date()`.

If there is a valid date string according to format *x*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error occurs before the end, `err` |= `ios_base::failbit`. If parsing reads all the characters, `err` |= `ios_base::eofbit`.

**Parameters**

<code>__beg</code>	Start of string to parse.
--------------------	---------------------------



<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

#### Returns

Iterator to first char beyond date string.

Definition at line 456 of file `locale_facets_nonio.h`.

References `std::time_get<_CharT, _InIter>::do_get_date()`.

```
4.698.4.9  template<typename _CharT, typename _InIter> iter_type std::time_get<_CharT, _InIter>::get_monthname (
            iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const  [inline]
```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_monthname()`.

Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

#### Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

#### Returns

Iterator to first char beyond month name.

Definition at line 513 of file `locale_facets_nonio.h`.

References `std::time_get<_CharT, _InIter>::do_get_monthname()`.

```
4.698.4.10 template<typename _CharT, typename _InIter> iter_type std::time_get<_CharT, _InIter>::get_time ( iter_type
            __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const  [inline]
```

Parse input time string.

This function parses a time according to the format *X* and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_time()`.

If there is a valid time string according to format *X*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

## Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

## Returns

Iterator to first char beyond time string.

Definition at line 431 of file locale\_facets\_nonio.h.

References std::time\_get< \_CharT, \_InIter >::do\_get\_time().

4.698.4.11 `template<typename _CharT, typename _InIter> iter_type std::time_get< _CharT, _InIter >::get_weekday ( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const [inline]`

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. The result is returned by calling time\_get::do\_get\_weekday().

Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, err |= ios\_base::failbit. If parsing reads all the characters, err |= ios\_base::eofbit.

## Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

## Returns

Iterator to first char beyond weekday name.

Definition at line 484 of file locale\_facets\_nonio.h.

References std::time\_get< \_CharT, \_InIter >::do\_get\_weekday().

4.698.4.12 `template<typename _CharT, typename _InIter> iter_type std::time_get< _CharT, _InIter >::get_year ( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const [inline]`

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling time\_get::do\_get\_year().

4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.

If an error occurs before the end, err |= ios\_base::failbit. If parsing reads all the characters, err |= ios\_base::eofbit.

## Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

## Returns

Iterator to first char beyond year.

Definition at line 539 of file `locale_facets_nonio.h`.

References `std::time_get<_CharT, _InIter>::do_get_year()`.

## 4.698.5 Member Data Documentation

4.698.5.1 `template<typename _CharT, typename _InIter> locale::id std::time_get<_CharT, _InIter>::id` `[static]`

Numpunct facet id.

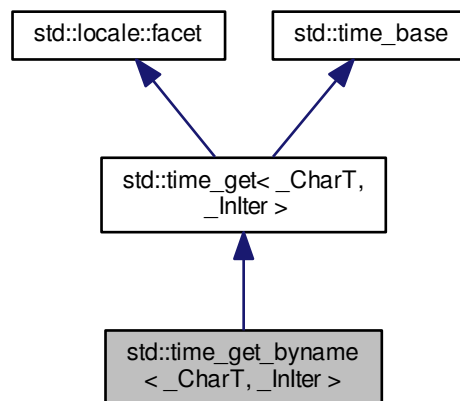
Definition at line 380 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

4.699 `std::time_get_byname<_CharT, _InIter>` Class Template Reference

Inheritance diagram for `std::time_get_byname<_CharT, _InIter>`:



## Public Types

- typedef [basic\\_string](#)<\_CharT> [\\_\\_string\\_type](#)
- typedef \_CharT [char\\_type](#)
- enum [dateorder](#) { [no\\_order](#), [dmy](#), [mdy](#), [ymd](#), [ydm](#) }
- typedef \_InIter [iter\\_type](#)

## Public Member Functions

- [time\\_get\\_byname](#) (const char \*, size\_t \_\_refs=0)
- [dateorder](#) [date\\_order](#) () const
- [iter\\_type](#) [get\\_date](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- [iter\\_type](#) [get\\_monthname](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- [iter\\_type](#) [get\\_time](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- [iter\\_type](#) [get\\_weekday](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- [iter\\_type](#) [get\\_year](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const

## Static Public Attributes

- static [locale::id](#) [id](#)

## Protected Member Functions

- [iter\\_type](#) [M\\_extract\\_name](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, int &\_\_member, const \_CharT \*\*\_\_names, size\_t \_\_indexlen, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err) const
- [iter\\_type](#) [M\\_extract\\_num](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, int &\_\_member, int \_\_min, int \_\_max, size\_t \_\_len, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err) const
- [iter\\_type](#) [M\\_extract\\_via\\_format](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm, const \_CharT \*\_\_format) const
- [iter\\_type](#) [M\\_extract\\_wday\\_or\\_month](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, int &\_\_member, const \_CharT \*\*\_\_names, size\_t \_\_indexlen, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err) const
- virtual [dateorder](#) [do\\_date\\_order](#) () const
- virtual [iter\\_type](#) [do\\_get\\_date](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- virtual [iter\\_type](#) [do\\_get\\_monthname](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- virtual [iter\\_type](#) [do\\_get\\_time](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- virtual [iter\\_type](#) [do\\_get\\_weekday](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- virtual [iter\\_type](#) [do\\_get\\_year](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const

## Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)

- static `__c_locale` **S\_get\_c\_locale** ()
- static const char \* **S\_get\_c\_name** () throw ()
- static `__c_locale` **S\_lc\_ctype\_c\_locale** (`__c_locale` `__cloc`, const char \* `__s`)

#### 4.699.1 Detailed Description

template<typename `_CharT`, typename `_InIter`>class std::time\_get\_byname< `_CharT`, `_InIter` >

class time\_get\_byname [22.2.5.2].

Definition at line 686 of file locale\_facets\_nonio.h.

#### 4.699.2 Member Function Documentation

4.699.2.1 template<typename `_CharT`, typename `_InIter` > dateorder std::time\_get< `_CharT`, `_InIter` >::date\_order ( ) const  
[inline], [inherited]

Return preferred order of month, day, and year.

This function returns an enum from `timebase::dateorder` giving the preferred ordering if the format `x` given to `time_put::put()` only uses month, day, and year. If the format `x` for the associated locale uses other fields, this function returns `timebase::dateorder::noorder`.

NOTE: The library always returns `noorder` at the moment.

##### Returns

A member of `timebase::dateorder`.

Definition at line 407 of file locale\_facets\_nonio.h.

References `std::time_get< _CharT, _InIter >::do_date_order()`.

4.699.2.2 template<typename `_CharT`, typename `_InIter` > virtual dateorder std::time\_get< `_CharT`, `_InIter` >::do\_date\_order ( ) const  
[protected], [virtual], [inherited]

Return preferred order of month, day, and year.

This function returns an enum from `timebase::dateorder` giving the preferred ordering if the format `x` given to `time_put::put()` only uses month, day, and year. This function is a hook for derived classes to change the value returned.

##### Returns

A member of `timebase::dateorder`.

Referenced by `std::time_get< _CharT, _InIter >::date_order()`.

4.699.2.3 template<typename `_CharT`, typename `_InIter` > virtual iter\_type std::time\_get< `_CharT`, `_InIter` >::do\_get\_date ( iter\_type `__beg`, iter\_type `__end`, ios\_base & `__io`, ios\_base::iostate & `__err`, tm \* `__tm` ) const  
[protected], [virtual], [inherited]

Parse input date string.

This function parses a date according to the format `X` and puts the results into a user-supplied struct `tm`. This function is a hook for derived classes to change the value returned.

See also

get\_date() for details.

Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond date string.

Referenced by std::time\_get< \_CharT, \_InIter >::get\_date().

```
4.699.2.4 template<typename _CharT, typename _InIter> virtual iter_type std::time_get< _CharT, _InIter
>::do_get_monthname( iter_type __beg, iter_type __end, ios_base &, ios_base::iostate & __err, tm * __tm )
const [protected], [virtual], [inherited]
```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get\_monthname() for details.

Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond month name.

Referenced by std::time\_get< \_CharT, \_InIter >::get\_monthname().

```
4.699.2.5 template<typename _CharT, typename _InIter> virtual iter_type std::time_get< _CharT, _InIter>::do_get_time
( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const
[protected], [virtual], [inherited]
```

Parse input time string.

This function parses a time according to the format x and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get\_time() for details.

**Parameters**

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

**Returns**

Iterator to first char beyond time string.

Referenced by `std::time_get<_CharT, _InIter>::get_time()`.

```
4.699.2.6 template<typename _CharT, typename _InIter> virtual iter_type std::time_get<_CharT, _InIter>::do_get_weekday
( iter_type __beg, iter_type __end, ios_base &, ios_base::iostate & __err, tm * __tm ) const
[protected], [virtual], [inherited]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

**See also**

`get_weekday()` for details.

**Parameters**

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

**Returns**

Iterator to first char beyond weekday name.

Referenced by `std::time_get<_CharT, _InIter>::get_weekday()`.

```
4.699.2.7 template<typename _CharT, typename _InIter> virtual iter_type std::time_get<_CharT, _InIter>::do_get_year
( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const
[protected], [virtual], [inherited]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

**See also**

`get_year()` for details.

## Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

## Returns

Iterator to first char beyond year.

Referenced by `std::time_get< _CharT, _InIter >::get_year()`.

**4.699.2.8** `template<typename _CharT, typename _InIter> iter_type std::time_get< _CharT, _InIter >::get_date ( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const [inline], [inherited]`

Parse input date string.

This function parses a date according to the format *x* and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_date()`.

If there is a valid date string according to format *x*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

## Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

## Returns

Iterator to first char beyond date string.

Definition at line 456 of file `locale_facets_nonio.h`.

References `std::time_get< _CharT, _InIter >::do_get_date()`.

**4.699.2.9** `template<typename _CharT, typename _InIter> iter_type std::time_get< _CharT, _InIter >::get_monthname ( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const [inline], [inherited]`

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_monthname()`.

Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.



**Parameters**

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

**Returns**

Iterator to first char beyond month name.

Definition at line 513 of file locale\_facets\_nonio.h.

References `std::time_get<_CharT, _InIter>::do_get_monthname()`.

```
4.699.2.10 template<typename _CharT, typename _InIter> iter_type std::time_get<_CharT, _InIter>::get_time ( iter_type
    __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const [inline],
    [inherited]
```

Parse input time string.

This function parses a time according to the format *X* and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_time()`.

If there is a valid time string according to format *X*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

**Parameters**

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

**Returns**

Iterator to first char beyond time string.

Definition at line 431 of file locale\_facets\_nonio.h.

References `std::time_get<_CharT, _InIter>::do_get_time()`.

```
4.699.2.11 template<typename _CharT, typename _InIter> iter_type std::time_get<_CharT, _InIter>::get_weekday
    ( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const
    [inline], [inherited]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_weekday()`.

Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

## Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

## Returns

Iterator to first char beyond weekday name.

Definition at line 484 of file locale\_facets\_nonio.h.

References `std::time_get< _CharT, _InIter >::do_get_weekday()`.

**4.699.2.12** `template<typename _CharT, typename _InIter> iter_type std::time_get< _CharT, _InIter >::get_year ( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const` `[inline]`,  
`[inherited]`

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_year()`.

4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.

If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

## Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

## Returns

Iterator to first char beyond year.

Definition at line 539 of file locale\_facets\_nonio.h.

References `std::time_get< _CharT, _InIter >::do_get_year()`.

**4.699.3 Member Data Documentation**

**4.699.3.1** `template<typename _CharT, typename _InIter> locale::id std::time_get< _CharT, _InIter >::id` `[static]`,  
`[inherited]`

Numpunct facet id.

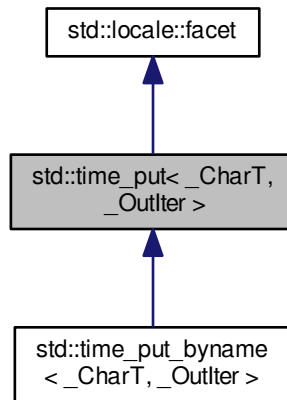
Definition at line 380 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

#### 4.700 `std::time_put<_CharT, _Outlter>` Class Template Reference

Inheritance diagram for `std::time_put<_CharT, _Outlter>`:



##### Public Types

- typedef `_CharT` `char_type`
- typedef `_Outlter` `iter_type`

##### Public Member Functions

- `time_put` (`size_t` \_\_refs=0)
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_io, `char_type` \_\_fill, const `tm` \*\_\_tm, const `_CharT` \*\_\_beg, const `_CharT` \*\_\_end) const
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_io, `char_type` \_\_fill, const `tm` \*\_\_tm, `char` \_\_format, `char` \_\_mod=0) const

##### Static Public Attributes

- static `locale::id` id

##### Protected Member Functions

- virtual `~time_put` ()
- virtual `iter_type do_put` (`iter_type` \_\_s, `ios_base` &\_\_io, `char_type` \_\_fill, const `tm` \*\_\_tm, `char` \_\_format, `char` \_\_mod) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## 4.700.1 Detailed Description

template<typename \_CharT, typename \_Outiter>class std::time\_put<\_CharT, \_Outiter>

Primary class template time\_put.

This facet encapsulates the code to format and output dates and times according to formats used by strftime().

The time\_put template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the time\_put facet.

Definition at line 715 of file locale\_facets\_nonio.h.

## 4.700.2 Member Typedef Documentation

4.700.2.1 template<typename \_CharT, typename \_Outiter> typedef \_CharT std::time\_put<\_CharT, \_Outiter>::char\_type

Public typedefs.

Definition at line 721 of file locale\_facets\_nonio.h.

4.700.2.2 template<typename \_CharT, typename \_Outiter> typedef \_Outiter std::time\_put<\_CharT, \_Outiter>::iter\_type

Public typedefs.

Definition at line 722 of file locale\_facets\_nonio.h.

## 4.700.3 Constructor &amp; Destructor Documentation

4.700.3.1 template<typename \_CharT, typename \_Outiter> std::time\_put<\_CharT, \_Outiter>::time\_put ( size\_t \_\_refs = 0 )  
[inline], [explicit]

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

<b>__refs</b>	Passed to the base facet class.
---------------	---------------------------------

Definition at line 736 of file locale\_facets\_nonio.h.

4.700.3.2 template<typename \_CharT, typename \_Outiter> virtual std::time\_put<\_CharT, \_Outiter>::~~time\_put ( )  
[inline], [protected], [virtual]

Destructor.

Definition at line 782 of file locale\_facets\_nonio.h.

#### 4.700.4 Member Function Documentation

4.700.4.1 `template<typename _CharT, typename _Outlter > virtual iter_type std::time_put< _CharT, _Outlter >::do_put ( iter_type __s, ios_base & __io, char_type __fill, const tm * __tm, char __format, char __mod ) const`  
`[protected], [virtual]`

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

See also

`put()` for more details.

##### Parameters

<code>__s</code>	The stream to write to.
<code>__io</code>	Source of locale.
<code>__fill</code>	char_type to use for padding.
<code>__tm</code>	Struct tm with date and time info to format.
<code>__format</code>	Format char.
<code>__mod</code>	Optional modifier char.

##### Returns

Iterator after writing.

Referenced by `std::time_put< _CharT, _Outlter >::put()`.

4.700.4.2 `template<typename _CharT, typename _Outlter > iter_type std::time_put< _CharT, _Outlter >::put ( iter_type __s, ios_base & __io, char_type __fill, const tm * __tm, const _CharT * __beg, const _CharT * __end ) const`

Format and output a time or date.

This function formats the data in struct tm according to the provided format string. The format string is interpreted as by `strftime()`.

##### Parameters

<code>__s</code>	The stream to write to.
<code>__io</code>	Source of locale.
<code>__fill</code>	char_type to use for padding.
<code>__tm</code>	Struct tm with date and time info to format.
<code>__beg</code>	Start of format string.
<code>__end</code>	End of format string.

##### Returns

Iterator after writing.

4.700.4.3 `template<typename _CharT, typename _Outlter > iter_type std::time_put< _CharT, _Outlter >::put ( iter_type __s, ios_base & __io, char_type __fill, const tm * __tm, char __format, char __mod = 0 ) const` `[inline]`

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. The format and modifier are interpreted as by `strftime()`. It does so by returning `time_put::do_put()`.

## Parameters

<code>__s</code>	The stream to write to.
<code>__io</code>	Source of locale.
<code>__fill</code>	char_type to use for padding.
<code>__tm</code>	Struct tm with date and time info to format.
<code>__format</code>	Format char.
<code>__mod</code>	Optional modifier char.

## Returns

Iterator after writing.

Definition at line 775 of file locale\_facets\_nonio.h.

References `std::time_put< _CharT, _Outlter >::do_put()`.

## 4.700.5 Member Data Documentation

4.700.5.1 `template<typename _CharT, typename _Outlter> locale::id std::time_put< _CharT, _Outlter >::id` `[static]`

Numpunct facet id.

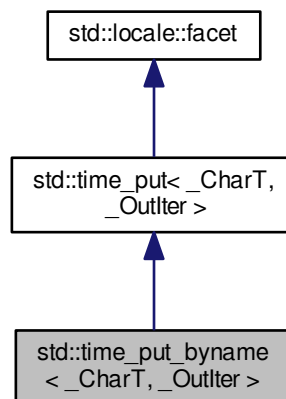
Definition at line 726 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 4.701 std::time\_put\_byname&lt; \_CharT, \_Outlter &gt; Class Template Reference

Inheritance diagram for `std::time_put_byname< _CharT, _Outlter >`:



**Public Types**

- typedef `_CharT` **char\_type**
- typedef `_Outlter` **iter\_type**

**Public Member Functions**

- **time\_put\_byname** (const char \*, size\_t \_\_refs=0)
- **iter\_type put** (iter\_type \_\_s, ios\_base &\_\_io, char\_type \_\_fill, const tm \*\_\_tm, const \_CharT \*\_\_beg, const \_CharT \*\_\_end) const
- **iter\_type put** (iter\_type \_\_s, ios\_base &\_\_io, char\_type \_\_fill, const tm \*\_\_tm, char \_\_format, char \_\_mod=0) const

**Static Public Attributes**

- static `locale::id` **id**

**Protected Member Functions**

- virtual **iter\_type do\_put** (iter\_type \_\_s, ios\_base &\_\_io, char\_type \_\_fill, const tm \*\_\_tm, char \_\_format, char \_\_mod) const

**Static Protected Member Functions**

- static `_c_locale` **\_S\_clone\_c\_locale** (`_c_locale` &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (`_c_locale` &\_\_cloc, const char \*\_\_s, `_c_locale` \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (`_c_locale` &\_\_cloc)
- static `_c_locale` **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static `_c_locale` **\_S\_lc\_ctype\_c\_locale** (`_c_locale` \_\_cloc, const char \*\_\_s)

**4.701.1 Detailed Description**

```
template<typename _CharT, typename _Outlter>class std::time_put_byname< _CharT, _Outlter >
```

class time\_put\_byname [22.2.5.4].

Definition at line 811 of file locale\_facets\_nonio.h.

**4.701.2 Member Function Documentation**

**4.701.2.1** `template<typename _CharT, typename _Outlter > virtual iter_type std::time_put< _CharT, _Outlter >::do_put (iter_type __s, ios_base & __io, char_type __fill, const tm *__tm, char __format, char __mod ) const`  
 [protected], [virtual], [inherited]

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

See also

put() for more details.

#### Parameters

<code>__s</code>	The stream to write to.
<code>__io</code>	Source of locale.
<code>__fill</code>	char_type to use for padding.
<code>__tm</code>	Struct tm with date and time info to format.
<code>__format</code>	Format char.
<code>__mod</code>	Optional modifier char.

#### Returns

Iterator after writing.

Referenced by std::time\_put< \_CharT, \_Outlter >::put().

**4.701.2.2** `template<typename _CharT, typename _Outlter> iter_type std::time_put< _CharT, _Outlter >::put ( iter_type __s, ios_base & __io, char_type __fill, const tm * __tm, const _CharT * __beg, const _CharT * __end ) const` [inherited]

Format and output a time or date.

This function formats the data in struct tm according to the provided format string. The format string is interpreted as by strftime().

#### Parameters

<code>__s</code>	The stream to write to.
<code>__io</code>	Source of locale.
<code>__fill</code>	char_type to use for padding.
<code>__tm</code>	Struct tm with date and time info to format.
<code>__beg</code>	Start of format string.
<code>__end</code>	End of format string.

#### Returns

Iterator after writing.

**4.701.2.3** `template<typename _CharT, typename _Outlter> iter_type std::time_put< _CharT, _Outlter >::put ( iter_type __s, ios_base & __io, char_type __fill, const tm * __tm, char __format, char __mod = 0 ) const` [inline], [inherited]

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. The format and modifier are interpreted as by strftime(). It does so by returning time\_put::do\_put().

#### Parameters

<code>__s</code>	The stream to write to.
------------------	-------------------------



<code>__io</code>	Source of locale.
<code>__fill</code>	<code>char_type</code> to use for padding.
<code>__tm</code>	Struct <code>tm</code> with date and time info to format.
<code>__format</code>	Format char.
<code>__mod</code>	Optional modifier char.

#### Returns

Iterator after writing.

Definition at line 775 of file `locale_facets_nonio.h`.

References `std::time_put< _CharT, _Outlter >::do_put()`.

#### 4.701.3 Member Data Documentation

4.701.3.1 `template<typename _CharT, typename _Outlter > locale::id std::time_put< _CharT, _Outlter >::id` `[static]`, `[inherited]`

Numpunct facet id.

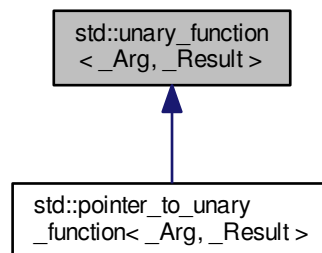
Definition at line 726 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

#### 4.702 `std::unary_function< _Arg, _Result >` Struct Template Reference

Inheritance diagram for `std::unary_function< _Arg, _Result >`:



#### Public Types

- typedef `_Arg` [argument\\_type](#)
- typedef `_Result` [result\\_type](#)

## 4.702.1 Detailed Description

```
template<typename _Arg, typename _Result> struct std::unary_function< _Arg, _Result >
```

This is one of the [functor base classes](#).

Definition at line 101 of file `stl_function.h`.

## 4.702.2 Member Typedef Documentation

4.702.2.1 `template<typename _Arg, typename _Result> typedef _Arg std::unary_function< _Arg, _Result >::argument_type`

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

4.702.2.2 `template<typename _Arg, typename _Result> typedef _Result std::unary_function< _Arg, _Result >::result_type`

`result_type` is the return type

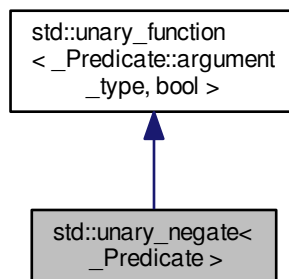
Definition at line 107 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.703 std::unary\_negate&lt; \_Predicate &gt; Class Template Reference

Inheritance diagram for `std::unary_negate< _Predicate >`:



## Public Types

- typedef `_Predicate::argument_type` [argument\\_type](#)
- typedef `bool` [result\\_type](#)

## Public Member Functions

- **unary\_negate** (const `_Predicate` &\_\_x)
- bool **operator()** (const typename `_Predicate::argument_type` &\_\_x) const

## Protected Attributes

- `_Predicate` **\_M\_pred**

## 4.703.1 Detailed Description

```
template<typename _Predicate>class std::unary_negate< _Predicate >
```

One of the [negation functors](#).

Definition at line 351 of file `stl_function.h`.

## 4.703.2 Member Typedef Documentation

4.703.2.1 `typedef _Predicate::argument_type std::unary_function< _Predicate::argument_type, bool >::argument_type` [\[inherited\]](#)

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

4.703.2.2 `typedef bool std::unary_function< _Predicate::argument_type, bool >::result_type` [\[inherited\]](#)

`result_type` is the return type

Definition at line 107 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

4.704 `std::uniform_int_distribution< _IntType >` Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- `typedef _IntType` [result\\_type](#)

## Public Member Functions

- [uniform\\_int\\_distribution](#) (`_IntType` \_\_a=0, `_IntType` \_\_b=std::numeric\_limits< `_IntType` >::max())
- **uniform\_int\_distribution** (const [param\\_type](#) &\_\_p)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator > void` **generate** (`_ForwardIterator` \_\_f, `_ForwardIterator` \_\_t, `_UniformRandomNumberGenerator` &\_\_urng)

- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator> void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator> void **\_\_generate** ([result\\_type](#) \* \_\_f, [result\\_type](#) \* \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) **a** () const
- [result\\_type](#) **b** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator> [result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator> [result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

#### Friends

- bool **operator==** (const [uniform\\_int\\_distribution](#) &\_\_d1, const [uniform\\_int\\_distribution](#) &\_\_d2)

#### 4.704.1 Detailed Description

template<typename \_IntType = int> class std::uniform\_int\_distribution< \_IntType >

Uniform discrete distribution for random numbers. A discrete random distribution on the range  $[min, max]$  with equal probability throughout the range.

Definition at line 1666 of file random.h.

#### 4.704.2 Member Typedef Documentation

4.704.2.1 template<typename \_IntType = int> typedef \_IntType std::uniform\_int\_distribution< \_IntType >::result\_type

The type of the range of the distribution.

Definition at line 1669 of file random.h.

#### 4.704.3 Constructor & Destructor Documentation

4.704.3.1 template<typename \_IntType = int> std::uniform\_int\_distribution< \_IntType >::uniform\_int\_distribution ( \_IntType \_\_a = 0, \_IntType \_\_b = std::numeric\_limits<\_IntType>::max() ) [inline], [explicit]

Constructs a uniform distribution object.

Definition at line 1709 of file random.h.

#### 4.704.4 Member Function Documentation

4.704.4.1 `template<typename _IntType = int> result_type std::uniform_int_distribution< _IntType >::max ( ) const`  
`[inline]`

Returns the inclusive upper bound of the distribution range.

Definition at line 1761 of file random.h.

4.704.4.2 `template<typename _IntType = int> result_type std::uniform_int_distribution< _IntType >::min ( ) const`  
`[inline]`

Returns the inclusive lower bound of the distribution range.

Definition at line 1754 of file random.h.

4.704.4.3 `template<typename _IntType = int> template<typename _UniformRandomNumberGenerator > result_type`  
`std::uniform_int_distribution< _IntType >::operator() ( _UniformRandomNumberGenerator & __urng )`  
`[inline]`

Generating functions.

Definition at line 1769 of file random.h.

4.704.4.4 `template<typename _IntType = int> param_type std::uniform_int_distribution< _IntType >::param ( ) const`  
`[inline]`

Returns the parameter set of the distribution.

Definition at line 1739 of file random.h.

4.704.4.5 `template<typename _IntType = int> void std::uniform_int_distribution< _IntType >::param ( const param_type`  
`& __param ) [inline]`

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 1747 of file random.h.

4.704.4.6 `template<typename _IntType = int> void std::uniform_int_distribution< _IntType >::reset ( ) [inline]`

Resets the distribution state.

Does nothing for the uniform integer distribution.

Definition at line 1725 of file random.h.

#### 4.704.5 Friends And Related Function Documentation

4.704.5.1 `template<typename _IntType = int> bool operator==( const uniform_int_distribution< _IntType > & __d1, const`  
`uniform_int_distribution< _IntType > & __d2 ) [friend]`

Return true if two uniform integer distributions have the same parameters.

Definition at line 1804 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

4.705 `std::uniform_int_distribution< _IntType >::param_type` Struct Reference

## Public Types

- typedef `uniform_int_distribution< _IntType >` **distribution\_type**

## Public Member Functions

- **param\_type** (`_IntType __a=0, _IntType __b=std::numeric_limits< _IntType >::max()`)
- **result\_type a** () const
- **result\_type b** () const

## Friends

- bool **operator==** (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

## 4.705.1 Detailed Description

template<typename `_IntType` = int>struct `std::uniform_int_distribution< _IntType >::param_type`

Parameter type.

Definition at line 1675 of file `random.h`.

The documentation for this struct was generated from the following file:

- `random.h`

4.706 `std::uniform_real_distribution< _RealType >` Class Template Reference

## Classes

- struct `param_type`

## Public Types

- typedef `_RealType` **result\_type**

## Public Member Functions

- `uniform_real_distribution` (`_RealType __a=_RealType(0), _RealType __b=_RealType(1)`)
- **uniform\_real\_distribution** (const `param_type` &\_\_p)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` > void **\_\_generate** (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng`)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` > void **\_\_generate** (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p`)
- template<typename `_UniformRandomNumberGenerator` > void **\_\_generate** (`result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p`)
- **result\_type a** () const
- **result\_type b** () const

- `result_type max () const`
- `result_type min () const`
- `template<typename _UniformRandomNumberGenerator > result_type operator() (_UniformRandomNumberGenerator && __urng)`
- `template<typename _UniformRandomNumberGenerator > result_type operator() (_UniformRandomNumberGenerator && __urng, const param_type &__p)`
- `param_type param () const`
- `void param (const param_type &__param)`
- `void reset ()`

#### Friends

- `bool operator== (const uniform_real_distribution &__d1, const uniform_real_distribution &__d2)`

#### 4.706.1 Detailed Description

`template<typename _RealType = double> class std::uniform_real_distribution< _RealType >`

Uniform continuous distribution for random numbers.

A continuous random distribution on the range [min, max) with equal probability throughout the range. The URNG should be real-valued and deliver number in the range [0, 1).

Definition at line 1867 of file random.h.

#### 4.706.2 Member Typedef Documentation

4.706.2.1 `template<typename _RealType = double> typedef _RealType std::uniform_real_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 1870 of file random.h.

#### 4.706.3 Constructor & Destructor Documentation

4.706.3.1 `template<typename _RealType = double> std::uniform_real_distribution< _RealType >::uniform_real_distribution ( _RealType __a = _RealType(0), _RealType __b = _RealType(1) )`  
`[inline], [explicit]`

Constructs a uniform\_real\_distribution object.

##### Parameters

<code>__a</code>	[IN] The lower bound of the distribution.
<code>__b</code>	[IN] The upper bound of the distribution.

Definition at line 1913 of file random.h.

#### 4.706.4 Member Function Documentation

4.706.4.1 `template<typename _RealType = double> result_type std::uniform_real_distribution<_RealType>::max ( )`  
`const [inline]`

Returns the inclusive upper bound of the distribution range.

Definition at line 1965 of file random.h.

4.706.4.2 `template<typename _RealType = double> result_type std::uniform_real_distribution<_RealType>::min ( )`  
`const [inline]`

Returns the inclusive lower bound of the distribution range.

Definition at line 1958 of file random.h.

4.706.4.3 `template<typename _RealType = double> template<typename UniformRandomNumberGenerator> result_type`  
`std::uniform_real_distribution<_RealType>::operator() ( UniformRandomNumberGenerator & __urng )`  
`[inline]`

Generating functions.

Definition at line 1973 of file random.h.

4.706.4.4 `template<typename _RealType = double> param_type std::uniform_real_distribution<_RealType>::param ( )`  
`const [inline]`

Returns the parameter set of the distribution.

Definition at line 1943 of file random.h.

4.706.4.5 `template<typename _RealType = double> void std::uniform_real_distribution<_RealType>::param ( const`  
`param_type & __param ) [inline]`

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 1951 of file random.h.

4.706.4.6 `template<typename _RealType = double> void std::uniform_real_distribution<_RealType>::reset ( )`  
`[inline]`

Resets the distribution state.

Does nothing for the uniform real distribution.

Definition at line 1929 of file random.h.

#### 4.706.5 Friends And Related Function Documentation

4.706.5.1 `template<typename _RealType = double> bool operator==( const uniform_real_distribution<_RealType> & __d1,`  
`const uniform_real_distribution<_RealType> & __d2 ) [friend]`

Return true if two uniform real distributions have the same parameters.

Definition at line 2013 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)



#### 4.707 `std::uniform_real_distribution<_RealType>::param_type` Struct Reference

##### Public Types

- typedef `uniform_real_distribution<_RealType>` **distribution\_type**

##### Public Member Functions

- **param\_type** (`_RealType __a=_RealType(0), _RealType __b=_RealType(1)`)
- **result\_type a** () const
- **result\_type b** () const

##### Friends

- bool **operator==** (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

##### 4.707.1 Detailed Description

template<typename `_RealType` = double>struct std::uniform\_real\_distribution<\_RealType>::param\_type

Parameter type.

Definition at line 1876 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

#### 4.708 `std::unique_ptr<_Tp, _Dp>` Class Template Reference

##### Public Types

- typedef `_Dp` **deleter\_type**
- typedef `_Tp` **element\_type**
- typedef `_Pointer::type` **pointer**

##### Public Member Functions

- **unique\_ptr** (`pointer __p`) noexcept
- **unique\_ptr** (`pointer __p, typename conditional< is_reference< deleter_type >::value, deleter_type, const deleter_type & >::type __d`) noexcept
- **unique\_ptr** (`pointer __p, typename remove_reference< deleter_type >::type &&__d`) noexcept
- constexpr **unique\_ptr** (`nullptr_t`) noexcept
- **unique\_ptr** (`unique_ptr &&__u`) noexcept
- template<typename `_Up`, typename `_Ep`, typename = `_Require< is_convertible<typename unique_ptr<_Up, _Ep>::pointer, pointer>, __not_<is_array<_Up>>, typename conditional<is_reference<_Dp>::value, is_same<_Ep, _Dp>, is_convertible<_Ep, _Dp>>::type>>` **unique\_ptr** (`unique_ptr<_Up, _Ep> &&__u`) noexcept
- **unique\_ptr** (`const unique_ptr &`)=delete
- template<typename `_Up`, typename `>` **unique\_ptr** (`auto_ptr<_Up> &&__u`) noexcept
- **pointer get** () const noexcept

- deleter\_type & **get\_deleter** () noexcept
- const deleter\_type & **get\_deleter** () const noexcept
- **operator bool** () const noexcept
- add\_lvalue\_reference< element\_type >::type **operator\*** () const
- pointer **operator->** () const noexcept
- **unique\_ptr** & **operator=** (**unique\_ptr** &&\_\_u) noexcept
- template<typename \_Up, typename \_Ep > enable\_if< \_\_and< is\_convertible< typename **unique\_ptr**< \_Up, \_Ep >::pointer, pointer >, \_\_not< is\_array< \_Up > > >::value, **unique\_ptr** & >::type **operator=** (**unique\_ptr**< \_Up, \_Ep > &&\_\_u) noexcept
- **unique\_ptr** & **operator=** (nullptr\_t) noexcept
- **unique\_ptr** & **operator=** (const **unique\_ptr** &)=delete
- pointer **release** () noexcept
- void **reset** (pointer \_\_p=pointer()) noexcept
- void **swap** (**unique\_ptr** &\_\_u) noexcept

#### 4.708.1 Detailed Description

template<typename \_Tp, typename \_Dp = default\_delete<\_Tp>> class std::unique\_ptr< \_Tp, \_Dp >

20.7.1.2 unique\_ptr for single objects.

Definition at line 109 of file unique\_ptr.h.

The documentation for this class was generated from the following files:

- [unique\\_ptr.h](#)
- [auto\\_ptr.h](#)

## 4.709 std::unique\_ptr< \_Tp[], \_Dp > Class Template Reference

### Public Types

- typedef \_Dp **deleter\_type**
- typedef \_Tp **element\_type**
- typedef \_Pointer::type **pointer**

### Public Member Functions

- **unique\_ptr** (pointer \_\_p) noexcept
- template<typename \_Up, typename = \_Require<is\_pointer<pointer>, is\_convertible<\_Up\*, pointer>, \_\_is\_derived\_Tp<\_Up>>> **unique\_ptr** (\_Up \*\_\_p)=delete
- **unique\_ptr** (pointer \_\_p, typename conditional< is\_reference< deleter\_type >::value, deleter\_type, const deleter\_type & >::type \_\_d) noexcept
- **unique\_ptr** (pointer \_\_p, typename remove\_reference< deleter\_type >::type &&\_\_d) noexcept
- **unique\_ptr** (**unique\_ptr** &&\_\_u) noexcept
- constexpr **unique\_ptr** (nullptr\_t) noexcept
- template<typename \_Up, typename \_Ep, typename = \_Require<\_\_safe\_conversion<\_Up, \_Ep>, typename conditional<is\_reference<\_\_<\_Dp>::value, is\_same<\_Ep, \_Dp>, is\_convertible<\_Ep, \_Dp>>::type >> **unique\_ptr** (**unique\_ptr**< \_Up, \_Ep > &&\_\_u) noexcept
- **unique\_ptr** (const **unique\_ptr** &)=delete

- `template<typename _Up, typename = _Require<is_pointer<pointer>, is_convertible<_Up*, pointer>, __is_derived_Tp<_Up>>>> unique_ptr (_Up *, typename conditional< is_reference< deleter_type >::value, deleter_type, const deleter_↵ type & >::type)=delete`
- `template<typename _Up, typename = _Require<is_pointer<pointer>, is_convertible<_Up*, pointer>, __is_derived_Tp<_Up>>>> unique_ptr (_Up *, typename remove_reference< deleter_type >::type &&)=delete`
- `pointer get () const noexcept`
- `deleter_type & get_deleter () noexcept`
- `const deleter_type & get_deleter () const noexcept`
- `operator bool () const noexcept`
- `unique_ptr & operator= (unique_ptr &&__u) noexcept`
- `template<typename _Up, typename _Ep > enable_if< __safe_conversion< _Up, _Ep >::value, unique_ptr & >::type operator= (unique_ptr< _Up, _Ep > &&__u) noexcept`
- `unique_ptr & operator= (nullptr_t) noexcept`
- `unique_ptr & operator= (const unique_ptr &)=delete`
- `std::add_lvalue_reference< element_type >::type operator[] (size_t __i) const`
- `pointer release () noexcept`
- `void reset () noexcept`
- `void reset (pointer __p) noexcept`
- `template<typename _Up, typename = _Require<is_pointer<pointer>, is_convertible<_Up*, pointer>, __is_derived_Tp<_Up>>>> void reset (_Up *)=delete`
- `void swap (unique_ptr &__u) noexcept`

#### 4.709.1 Detailed Description

`template<typename _Tp, typename _Dp>class std::unique_ptr< _Tp[], _Dp >`

##### 20.7.1.3 unique\_ptr for array objects with a runtime length

Definition at line 282 of file `unique_ptr.h`.

The documentation for this class was generated from the following file:

- [unique\\_ptr.h](#)

#### 4.710 std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc > Class Template Reference

Inherits `std::__allow_copy_cons< bool >`.

##### Public Types

- `typedef _Hashtable::key_type key\_type`
- `typedef _Hashtable::value_type value\_type`
- `typedef _Hashtable::mapped_type mapped\_type`
- `typedef _Hashtable::hasher hasher`
- `typedef _Hashtable::key_equal key\_equal`
- `typedef _Hashtable::allocator_type allocator\_type`
- `typedef allocator_type::pointer pointer`
- `typedef allocator_type::const_pointer const\_pointer`
- `typedef allocator_type::reference reference`
- `typedef allocator_type::const_reference const\_reference`

- typedef \_Hashtable::iterator [iterator](#)
- typedef \_Hashtable::const\_iterator [const\\_iterator](#)
- typedef \_Hashtable::local\_iterator [local\\_iterator](#)
- typedef \_Hashtable::const\_local\_iterator [const\\_local\\_iterator](#)
- typedef \_Hashtable::size\_type [size\\_type](#)
- typedef \_Hashtable::difference\_type [difference\\_type](#)

#### Public Member Functions

- [unordered\\_map](#) ([size\\_type](#) \_\_n=10, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- template<typename \_InputIterator > [unordered\\_map](#) (\_InputIterator \_\_f, \_InputIterator \_\_l, [size\\_type](#) \_\_n=0, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [unordered\\_map](#) (const [unordered\\_map](#) &)=default
- [unordered\\_map](#) ([unordered\\_map](#) &&)=default
- [unordered\\_map](#) (initializer\_list< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n=0, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [iterator](#) [begin](#) () noexcept
- [local\\_iterator](#) [begin](#) ([size\\_type](#) \_\_n)
- [size\\_type](#) [bucket](#) (const [key\\_type](#) &\_\_key) const
- [size\\_type](#) [bucket\\_count](#) () const noexcept
- [size\\_type](#) [bucket\\_size](#) ([size\\_type](#) \_\_n) const
- void [clear](#) () noexcept
- [size\\_type](#) [count](#) (const [key\\_type](#) &\_\_x) const
- template<typename... \_Args> [std::pair](#)< [iterator](#), bool > [emplace](#) (\_Args &&... \_\_args)
- template<typename... \_Args> [iterator](#) [emplace\\_hint](#) (const [iterator](#) \_\_pos, \_Args &&... \_\_args)
- bool [empty](#) () const noexcept
- [iterator](#) [end](#) () noexcept
- [local\\_iterator](#) [end](#) ([size\\_type](#) \_\_n)
- [size\\_type](#) [erase](#) (const [key\\_type](#) &\_\_x)
- [iterator](#) [erase](#) (const [iterator](#) \_\_first, const [iterator](#) \_\_last)
- [allocator\\_type](#) [get\\_allocator](#) () const noexcept
- [hasher](#) [hash\\_function](#) () const
- template<typename \_InputIterator > void [insert](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void [insert](#) (initializer\_list< [value\\_type](#) > \_\_l)
- [key\\_equal](#) [key\\_eq](#) () const
- float [load\\_factor](#) () const noexcept
- [size\\_type](#) [max\\_bucket\\_count](#) () const noexcept
- float [max\\_load\\_factor](#) () const noexcept
- void [max\\_load\\_factor](#) (float \_\_z)
- [size\\_type](#) [max\\_size](#) () const noexcept
- [unordered\\_map](#) & [operator=](#) (const [unordered\\_map](#) &)=default
- [unordered\\_map](#) & [operator=](#) ([unordered\\_map](#) &&)=default
- [unordered\\_map](#) & [operator=](#) (initializer\_list< [value\\_type](#) > \_\_l)
- void [rehash](#) ([size\\_type](#) \_\_n)
- void [reserve](#) ([size\\_type](#) \_\_n)
- [size\\_type](#) [size](#) () const noexcept
- void [swap](#) ([unordered\\_map](#) &\_\_x)
- [const\\_iterator](#) [begin](#) () const noexcept

- `const_iterator cbegin` () const noexcept
- `const_iterator end` () const noexcept
- `const_iterator cend` () const noexcept
- `std::pair< iterator, bool > insert` (const `value_type` &\_\_x)
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type> std::pair< iterator, bool > insert` (\_Pair &&\_\_x)
- `iterator insert` (const `iterator` \_\_hint, const `value_type` &\_\_x)
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type> iterator insert` (const `iterator` \_\_hint, \_Pair &&\_\_x)
- `iterator erase` (const `iterator` \_\_position)
- `iterator erase` (`iterator` \_\_it)
- `iterator find` (const `key_type` &\_\_x)
- `const_iterator find` (const `key_type` &\_\_x) const
- `std::pair< iterator, iterator > equal_range` (const `key_type` &\_\_x)
- `std::pair< const_iterator, const_iterator > equal_range` (const `key_type` &\_\_x) const
- `mapped_type & operator[]` (const `key_type` &\_\_k)
- `mapped_type & operator[]` (`key_type` &&\_\_k)
- `mapped_type & at` (const `key_type` &\_\_k)
- `const mapped_type & at` (const `key_type` &\_\_k) const
- `const_local_iterator begin` (`size_type` \_\_n) const
- `const_local_iterator cbegin` (`size_type` \_\_n) const
- `const_local_iterator end` (`size_type` \_\_n) const
- `const_local_iterator cend` (`size_type` \_\_n) const

## Friends

- `template<typename _Key1, typename _Tp1, typename _Hash1, typename _Pred1, typename _Alloc1 > bool operator==` (const `unordered_map< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 >` &, const `unordered_map< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 >` &)

## 4.710.1 Detailed Description

```
template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> class std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >
```

A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.

## Template Parameters

<code>_Key</code>	Type of key objects.
<code>_Tp</code>	Type of mapped objects.
<code>_Hash</code>	Hashing function object type, defaults to <code>hash&lt;_Value&gt;</code> .
<code>_Pred</code>	Predicate function object type, defaults to <code>equal_to&lt;_Value&gt;</code> .
<code>_Alloc</code>	Allocator type, defaults to <code>allocator&lt;_Key&gt;</code> .

Meets the requirements of a [container](#), and [unordered associative container](#)

The resulting value type of the container is `std::pair<const _Key, _Tp>`.

Base is `_Hashtable`, dispatched at compile time via template alias `__umap_hashtable`.

Definition at line 97 of file `unordered_map.h`.

## 4.710.2 Member Typedef Documentation

**4.710.2.1** `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::allocator_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::allocator_type`

Public typedefs.

Definition at line 111 of file `unordered_map.h`.

**4.710.2.2** `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::const_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::const_iterator`

Iterator-related typedefs.

Definition at line 121 of file `unordered_map.h`.

**4.710.2.3** `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::const_local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::const_local_iterator`

Iterator-related typedefs.

Definition at line 123 of file `unordered_map.h`.

**4.710.2.4** `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef allocator_type::const_pointer std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::const_pointer`

Iterator-related typedefs.

Definition at line 117 of file `unordered_map.h`.

**4.710.2.5** `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef allocator_type::const_reference std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::const_reference`

Iterator-related typedefs.

Definition at line 119 of file `unordered_map.h`.

4.710.2.6 `template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::difference_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::difference_type`

Iterator-related typedefs.

Definition at line 125 of file `unordered_map.h`.

4.710.2.7 `template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::hasher std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::hasher`

Public typedefs.

Definition at line 109 of file `unordered_map.h`.

4.710.2.8 `template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::iterator`

Iterator-related typedefs.

Definition at line 120 of file `unordered_map.h`.

4.710.2.9 `template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::key_equal std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::key_equal`

Public typedefs.

Definition at line 110 of file `unordered_map.h`.

4.710.2.10 `template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::key_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::key_type`

Public typedefs.

Definition at line 106 of file `unordered_map.h`.

4.710.2.11 `template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::local_iterator`

Iterator-related typedefs.

Definition at line 122 of file `unordered_map.h`.

4.710.2.12 `template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::mapped_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::mapped_type`

Public typedefs.

Definition at line 108 of file `unordered_map.h`.

4.710.2.13 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef allocator_type::pointer std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::pointer`

Iterator-related typedefs.

Definition at line 116 of file unordered\_map.h.

4.710.2.14 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef allocator_type::reference std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::reference`

Iterator-related typedefs.

Definition at line 118 of file unordered\_map.h.

4.710.2.15 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::size_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::size_type`

Iterator-related typedefs.

Definition at line 124 of file unordered\_map.h.

4.710.2.16 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::value_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::value_type`

Public typedefs.

Definition at line 107 of file unordered\_map.h.

#### 4.710.3 Constructor & Destructor Documentation

4.710.3.1 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::unordered_map ( size_type __n = 10, const hasher & __hf = hasher(), const key_equal & __eqf = key_equal(), const allocator_type & __a = allocator_type() ) [inline], [explicit]`

Default constructor creates no elements.

Parameters

<code>__n</code>	Initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eqf</code>	A key equality functor.
<code>__a</code>	An allocator object.

Definition at line 138 of file unordered\_map.h.

4.710.3.2 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> template<typename _InputIterator> std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::unordered_map ( _InputIterator __f, _InputIterator __l, size_type __n = 0, const hasher & __hf = hasher(), const key_equal & __eqf = key_equal(), const allocator_type & __a = allocator_type() ) [inline]`

Builds an unordered\_map from a range.



## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__n</code>	Minimal initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eqf</code>	A key equality functor.
<code>__a</code>	An allocator object.

Create an `unordered_map` consisting of copies of the elements from `[__first,__last)`. This is linear in  $N$  (where  $N$  is `distance(__first,__last)`).

Definition at line 159 of file `unordered_map.h`.

```
4.710.3.3  template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc
           = std::allocator<std::pair<const _Key, _Tp>>> std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
           >::unordered_map ( const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> & ) [default]
```

Copy constructor.

```
4.710.3.4  template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc
           = std::allocator<std::pair<const _Key, _Tp>>> std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
           >::unordered_map ( unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> && ) [default]
```

Move constructor.

```
4.710.3.5  template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc
           = std::allocator<std::pair<const _Key, _Tp>>> std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
           >::unordered_map ( initializer_list<value_type> & __l, size_type __n = 0, const hasher & __hf = hasher(),
           const key_equal & __eqf = key_equal(), const allocator_type & __a = allocator_type() ) [inline]
```

Builds an `unordered_map` from an `initializer_list`.

## Parameters

<code>__l</code>	An <code>initializer_list</code> .
<code>__n</code>	Minimal initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eqf</code>	A key equality functor.
<code>__a</code>	An allocator object.

Create an `unordered_map` consisting of copies of the elements in the list. This is linear in  $N$  (where  $N$  is `__l.size()`).

Definition at line 184 of file `unordered_map.h`.

## 4.710.4 Member Function Documentation

```
4.710.4.1  template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
           std::allocator<std::pair<const _Key, _Tp>>> mapped_type& std::unordered_map<_Key, _Tp, _Hash, _Pred,
           _Alloc>::at ( const key_type & __k ) [inline]
```

Access to `unordered_map` data.

## Parameters

<code>__k</code>	The key for which data should be retrieved.
------------------	---

**Returns**

A reference to the data whose key is equal to `__k`, if such a data is present in the `unordered_map`.

**Exceptions**

<code>std::out_of_range</code>	If no such data is present.
--------------------------------	-----------------------------

Definition at line 612 of file `unordered_map.h`.

```
4.710.4.2 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const mapped_type& std::unordered_map<_Key, _Tp, _Hash,
_Pred, _Alloc>::at ( const key_type & __k ) const [inline]
```

Access to `unordered_map` data.

**Parameters**

<code>__k</code>	The key for which data should be retrieved.
------------------	---

**Returns**

A reference to the data whose key is equal to `__k`, if such a data is present in the `unordered_map`.

**Exceptions**

<code>std::out_of_range</code>	If no such data is present.
--------------------------------	-----------------------------

Definition at line 616 of file `unordered_map.h`.

```
4.710.4.3 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::begin ( ) [inline], [noexcept]
```

Returns a read/write iterator that points to the first element in the `unordered_map`.

Definition at line 248 of file `unordered_map.h`.

```
4.710.4.4 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred,
_Alloc>::begin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the `unordered_map`.

Definition at line 257 of file `unordered_map.h`.

```
4.710.4.5 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred,
_Alloc>::begin ( size_type __n ) [inline]
```

Returns a read/write iterator pointing to the first bucket element.

**Parameters**

<code>__n</code>	The bucket index.
------------------	-------------------

**Returns**

A read/write local iterator.

Definition at line 657 of file `unordered_map.h`.

```
4.710.4.6  template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_local_iterator std::unordered_map<_Key, _Tp, _Hash,
_Pred, _Alloc>::begin ( size_type __n ) const    [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

<code>__n</code>	The bucket index.
------------------	-------------------

**Returns**

A read-only local iterator.

Definition at line 668 of file `unordered_map.h`.

```
4.710.4.7  template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::bucket_count ( ) const    [inline], [noexcept]
```

Returns the number of buckets of the `unordered_map`.

Definition at line 624 of file `unordered_map.h`.

```
4.710.4.8  template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred,
_Alloc>::cbegin ( ) const    [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the `unordered_map`.

Definition at line 261 of file `unordered_map.h`.

```
4.710.4.9  template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_local_iterator std::unordered_map<_Key, _Tp, _Hash,
_Pred, _Alloc>::cbegin ( size_type __n ) const    [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

<code>__n</code>	The bucket index.
------------------	-------------------

**Returns**

A read-only local iterator.

Definition at line 672 of file `unordered_map.h`.

4.710.4.10 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::cend( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the unordered\_map.

Definition at line 283 of file unordered\_map.h.

4.710.4.11 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::cend( size_type __n ) const [inline]`

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

<code>__n</code>	The bucket index.
------------------	-------------------

Returns

A read-only local iterator.

Definition at line 698 of file unordered\_map.h.

4.710.4.12 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::clear( ) [inline], [noexcept]`

Erases all elements in an unordered\_map. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 497 of file unordered\_map.h.

4.710.4.13 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::count( const key_type &__x ) const [inline]`

Finds the number of elements.

Parameters

<code>__x</code>	Key to count.
------------------	---------------

Returns

Number of elements with specified key.

This function only makes sense for unordered\_multimap; for unordered\_map the result will either be 0 (not present) or 1 (present).

Definition at line 560 of file unordered\_map.h.

4.710.4.14 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> template<typename... _Args> std::pair<iterator, bool> std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::emplace( _Args &&... __args ) [inline]`

Attempts to build and insert a std::pair into the unordered\_map.

## Parameters

<code>__args</code>	Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor).
---------------------	--

## Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to build and insert a (key, value) pair into the `unordered_map`. An `unordered_map` relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the `unordered_map`.

Insertion requires amortized constant time.

Definition at line 310 of file `unordered_map.h`.

```
4.710.4.15  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>>> template<typename... _Args> iterator std::unordered_map<
            _Key, _Tp, _Hash, _Pred, _Alloc >::emplace_hint ( const_iterator __pos, _Args &&... __args ) [inline]
```

Attempts to build and insert a `std::pair` into the `unordered_map`.

## Parameters

<code>__pos</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__args</code>	Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor).

## Returns

An iterator that points to the element with key of the `std::pair` built from `__args` (may or may not be that `std::pair`).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `emplace()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 340 of file `unordered_map.h`.

```
4.710.4.16  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>>> bool std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
            >::empty ( ) const [inline], [noexcept]
```

Returns true if the `unordered_map` is empty.

Definition at line 228 of file `unordered_map.h`.

```
4.710.4.17  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>>> iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
            >::end ( ) [inline], [noexcept]
```

Returns a read/write iterator that points one past the last element in the `unordered_map`.

Definition at line 270 of file `unordered_map.h`.

4.710.4.18 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::end( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the unordered\_map.

Definition at line 279 of file unordered\_map.h.

4.710.4.19 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::end( size_type __n ) [inline]`

Returns a read/write iterator pointing to one past the last bucket elements.

Parameters

<code>__n</code>	The bucket index.
------------------	-------------------

Returns

A read/write local iterator.

Definition at line 683 of file unordered\_map.h.

4.710.4.20 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::end( size_type __n ) const [inline]`

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

<code>__n</code>	The bucket index.
------------------	-------------------

Returns

A read-only local iterator.

Definition at line 694 of file unordered\_map.h.

4.710.4.21 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::pair<iterator, iterator> std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::equal_range( const key_type & __x ) [inline]`

Finds a subsequence matching given key.

Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for unordered\_multimap.

Definition at line 573 of file unordered\_map.h.

4.710.4.22 `template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class  
_Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::pair<const_iterator, const_iterator>  
std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::equal_range ( const key_type & __x ) const  
[inline]`

Finds a subsequence matching given key.

## Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

## Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for unordered\_multimap.

Definition at line 577 of file unordered\_map.h.

```
4.710.4.23 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::erase( const_iterator __position ) [inline]
```

Erases an element from an unordered\_map.

## Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

## Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered\_map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 447 of file unordered\_map.h.

```
4.710.4.24 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::erase( iterator __it ) [inline]
```

Erases an element from an unordered\_map.

## Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

## Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered\_map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 452 of file unordered\_map.h.

```
4.710.4.25 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::erase( const key_type & __x ) [inline]
```

Erases elements according to the provided key.



## Parameters

<code>__x</code>	Key of element to be erased.
------------------	------------------------------

## Returns

The number of elements erased.

This function erases all the elements located by the given key from an `unordered_map`. For an `unordered_map` the result of this function can only be 0 (not present) or 1 (present). Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 469 of file `unordered_map.h`.

```
4.710.4.26 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::erase( const_iterator __first, const_iterator __last ) [inline]
```

Erases a [`__first`,`__last`) range of elements from an `unordered_map`.

## Parameters

<code>__first</code>	Iterator pointing to the start of the range to be erased.
<code>__last</code>	Iterator pointing to the end of the range to be erased.

## Returns

The iterator `__last`.

This function erases a sequence of elements from an `unordered_map`. Note that this function only erases the elements, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 487 of file `unordered_map.h`.

```
4.710.4.27 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::find( const key_type &__x ) [inline]
```

Tries to locate an element in an `unordered_map`.

## Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

## Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

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

```
4.710.4.28 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred,
_Alloc>::find( const key_type &__x ) const [inline]
```

Tries to locate an element in an `unordered_map`.

## Parameters

__x	Key to be located.
-----	--------------------

## Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

Definition at line 546 of file unordered\_map.h.

```
4.710.4.29 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> allocator_type std::unordered_map<_Key, _Tp, _Hash, _Pred,
_Alloc>::get_allocator ( ) const [inline], [noexcept]
```

Returns the allocator object with which the unordered\_map was constructed.

Definition at line 221 of file unordered\_map.h.

```
4.710.4.30 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> hasher std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::hash_function ( ) const [inline]
```

Returns the hash functor object with which the unordered\_map was constructed.

Definition at line 518 of file unordered\_map.h.

```
4.710.4.31 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::pair<iterator, bool> std::unordered_map<_Key, _Tp,
_Hash, _Pred, _Alloc>::insert ( const value_type & __x ) [inline]
```

Attempts to insert a std::pair into the unordered\_map.

## Parameters

__x	Pair to be inserted (see std::make_pair for easy creation of pairs).
-----	--

## Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the unordered\_map. An unordered\_map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the unordered\_map.

Insertion requires amortized constant time.

Definition at line 362 of file unordered\_map.h.

```
4.710.4.32 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class
_Alloc = std::allocator<std::pair<const _Key, _Tp>>> template<typename _Pair, typename = typename
std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type> std::pair<iterator, bool>
std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::insert ( _Pair && __x ) [inline]
```

Attempts to insert a std::pair into the unordered\_map.

## Parameters

<code>__x</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).
------------------	---

## Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the `unordered_map`. An `unordered_map` relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the `unordered_map`.

Insertion requires amortized constant time.

Definition at line 369 of file `unordered_map.h`.

```
4.710.4.33 template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::insert ( const_iterator __hint, const value_type & __x ) [inline]
```

Attempts to insert a `std::pair` into the `unordered_map`.

## Parameters

<code>__hint</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__x</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).

## Returns

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 396 of file `unordered_map.h`.

```
4.710.4.34 template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class
_Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename _Pair , typename = typename
std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type> iterator std::unordered_map<_Key,
_Tp, _Hash, _Pred, _Alloc>::insert ( const_iterator __hint, _Pair && __x ) [inline]
```

Attempts to insert a `std::pair` into the `unordered_map`.

## Parameters

<code>__hint</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__x</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).

## Returns

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 403 of file `unordered_map.h`.

```
4.710.4.35 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> template<typename _InputIterator> void std::unordered_map<
_Key, _Tp, _Hash, _Pred, _Alloc>::insert( _InputIterator __first, _InputIterator __last ) [inline]
```

A template function that attempts to insert a range of elements.

## Parameters

<code>__first</code>	Iterator pointing to the start of the range to be inserted.
<code>__last</code>	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 418 of file `unordered_map.h`.

```
4.710.4.36 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>
::insert( initializer_list<value_type> __l ) [inline]
```

Attempts to insert a list of elements into the `unordered_map`.

## Parameters

<code>__l</code>	A <code>std::initializer_list&lt;value_type&gt;</code> of elements to be inserted.
------------------	--

Complexity similar to that of the range constructor.

Definition at line 429 of file `unordered_map.h`.

```
4.710.4.37 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> key_equal std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>
::key_eq( ) const [inline]
```

Returns the key comparison object with which the `unordered_map` was constructed.

Definition at line 524 of file `unordered_map.h`.

```
4.710.4.38 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> float std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>
::load_factor( ) const [inline], [noexcept]
```

Returns the average number of elements per bucket.

Definition at line 706 of file `unordered_map.h`.

4.710.4.39 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::max_bucket_count( ) const [inline], [noexcept]`

Returns the maximum number of buckets of the unordered\_map.

Definition at line 629 of file unordered\_map.h.

4.710.4.40 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> float std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::max_load_factor( ) const [inline], [noexcept]`

Returns a positive number that the unordered\_map tries to keep the load factor less than or equal to.

Definition at line 712 of file unordered\_map.h.

4.710.4.41 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::max_load_factor( float __z ) [inline]`

Change the unordered\_map maximum load factor.

Parameters

<code>__z</code>	The new maximum load factor.
------------------	------------------------------

Definition at line 720 of file unordered\_map.h.

4.710.4.42 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::max_size( ) const [inline], [noexcept]`

Returns the maximum size of the unordered\_map.

Definition at line 238 of file unordered\_map.h.

4.710.4.43 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> unordered_map& std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::operator=( const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> & ) [default]`

Copy assignment operator.

4.710.4.44 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> unordered_map& std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::operator=( unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> && ) [default]`

Move assignment operator.

4.710.4.45 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> unordered_map& std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::operator=( initializer_list<value_type> & ) [inline]`

Unordered\_map list assignment operator.

Parameters

<code>__l</code>	An initializer_list.
------------------	----------------------

This function fills an unordered\_map with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the unordered\_map and that the resulting unordered\_map's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 212 of file unordered\_map.h.

**4.710.4.46** `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> mapped_type& std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::operator[]( const key_type &__k ) [inline]`

Subscript ( `[]` ) access to unordered\_map data.

Parameters

<code>__k</code>	The key for which data should be retrieved.
------------------	---

Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ( `[]` ) operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.

Lookup requires constant time.

Definition at line 595 of file unordered\_map.h.

**4.710.4.47** `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> mapped_type& std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::operator[]( key_type &&__k ) [inline]`

Subscript ( `[]` ) access to unordered\_map data.

Parameters

<code>__k</code>	The key for which data should be retrieved.
------------------	---

Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ( `[]` ) operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.

Lookup requires constant time.

Definition at line 599 of file unordered\_map.h.

References `std::move()`.

**4.710.4.48** `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::rehash( size_type __n ) [inline]`

May rehash the unordered\_map.

## Parameters

<code>__n</code>	The new number of buckets.
------------------	----------------------------

Rehash will occur only if the new number of buckets respect the `unordered_map` maximum load factor.

Definition at line 731 of file `unordered_map.h`.

```
4.710.4.49 template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc
= std::allocator<std::pair<const _Key, _Tp> >> void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc
>::reserve ( size_type __n ) [inline]
```

Prepare the `unordered_map` for a specified number of elements.

## Parameters

<code>__n</code>	Number of elements required.
------------------	------------------------------

Same as `rehash(ceil(n / max_load_factor()))`.

Definition at line 742 of file `unordered_map.h`.

```
4.710.4.50 template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc
>::size ( ) const [inline], [noexcept]
```

Returns the size of the `unordered_map`.

Definition at line 233 of file `unordered_map.h`.

```
4.710.4.51 template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc
= std::allocator<std::pair<const _Key, _Tp> >> void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc
>::swap ( unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x ) [inline]
```

Swaps data with another `unordered_map`.

## Parameters

<code>__x</code>	An <code>unordered_map</code> of the same element and allocator types.
------------------	--

This exchanges the elements between two `unordered_map` in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

Definition at line 510 of file `unordered_map.h`.

The documentation for this class was generated from the following file:

- [unordered\\_map.h](#)

## 4.711 `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >` Class Template Reference

Inherits `std::__allow_copy_cons< bool >`.

## Public Types

- typedef `_Hashtable::key_type` [key\\_type](#)
- typedef `_Hashtable::value_type` [value\\_type](#)
- typedef `_Hashtable::mapped_type` [mapped\\_type](#)
- typedef `_Hashtable::hasher` [hasher](#)
- typedef `_Hashtable::key_equal` [key\\_equal](#)

- typedef \_Hashtable::allocator\_type [allocator\\_type](#)
- typedef allocator\_type::pointer [pointer](#)
- typedef allocator\_type::const\_pointer [const\\_pointer](#)
- typedef allocator\_type::reference [reference](#)
- typedef allocator\_type::const\_reference [const\\_reference](#)
- typedef \_Hashtable::iterator [iterator](#)
- typedef \_Hashtable::const\_iterator [const\\_iterator](#)
- typedef \_Hashtable::local\_iterator [local\\_iterator](#)
- typedef \_Hashtable::const\_local\_iterator [const\\_local\\_iterator](#)
- typedef \_Hashtable::size\_type [size\\_type](#)
- typedef \_Hashtable::difference\_type [difference\\_type](#)

### Public Member Functions

- [unordered\\_multimap](#) (size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator > [unordered\\_multimap](#) (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- [unordered\\_multimap](#) (const [unordered\\_multimap](#) &)=default
- [unordered\\_multimap](#) ([unordered\\_multimap](#) &&)=default
- [unordered\\_multimap](#) (initializer\_list< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- [iterator](#) begin () noexcept
- [local\\_iterator](#) begin (size\_type \_\_n)
- size\_type [bucket](#) (const key\_type &\_\_key) const
- size\_type [bucket\\_count](#) () const noexcept
- size\_type [bucket\\_size](#) (size\_type \_\_n) const
- void [clear](#) () noexcept
- size\_type [count](#) (const key\_type &\_\_x) const
- template<typename... \_Args> [iterator](#) [emplace](#) (\_Args &&...\_\_args)
- template<typename... \_Args> [iterator](#) [emplace\\_hint](#) (const\_iterator \_\_pos, \_Args &&...\_\_args)
- bool [empty](#) () const noexcept
- [iterator](#) end () noexcept
- [local\\_iterator](#) end (size\_type \_\_n)
- size\_type [erase](#) (const key\_type &\_\_x)
- [iterator](#) [erase](#) (const\_iterator \_\_first, const\_iterator \_\_last)
- allocator\_type [get\\_allocator](#) () const noexcept
- hasher [hash\\_function](#) () const
- template<typename \_InputIterator > void [insert](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void [insert](#) (initializer\_list< value\_type > \_\_l)
- key\_equal [key\\_eq](#) () const
- float [load\\_factor](#) () const noexcept
- size\_type [max\\_bucket\\_count](#) () const noexcept
- float [max\\_load\\_factor](#) () const noexcept
- void [max\\_load\\_factor](#) (float \_\_z)
- size\_type [max\\_size](#) () const noexcept
- [unordered\\_multimap](#) & [operator=](#) (const [unordered\\_multimap](#) &)=default
- [unordered\\_multimap](#) & [operator=](#) ([unordered\\_multimap](#) &&)=default
- [unordered\\_multimap](#) & [operator=](#) (initializer\_list< value\_type > \_\_l)



- void `rehash` (`size_type` \_\_n)
- void `reserve` (`size_type` \_\_n)
- `size_type` `size` () const noexcept
- void `swap` (`unordered_multimap` &\_\_x)
- `const_iterator` `begin` () const noexcept
- `const_iterator` `cbegin` () const noexcept
- `const_iterator` `end` () const noexcept
- `const_iterator` `cend` () const noexcept
- `iterator` `insert` (const `value_type` &\_\_x)
- template<typename `_Pair` , typename = typename std::enable\_if<std::is\_constructible<value\_type, `_Pair`&&>::value>::type> `iterator insert` (`_Pair` &&\_\_x)
- `iterator` `insert` (`const_iterator` \_\_hint, const `value_type` &\_\_x)
- template<typename `_Pair` , typename = typename std::enable\_if<std::is\_constructible<value\_type, `_Pair`&&>::value>::type> `iterator insert` (`const_iterator` \_\_hint, `_Pair` &&\_\_x)
- `iterator` `erase` (`const_iterator` \_\_position)
- `iterator` `erase` (`iterator` \_\_it)
- `iterator` `find` (const `key_type` &\_\_x)
- `const_iterator` `find` (const `key_type` &\_\_x) const
- `std::pair`< `iterator`, `iterator` > `equal_range` (const `key_type` &\_\_x)
- `std::pair`< `const_iterator`, `const_iterator` > `equal_range` (const `key_type` &\_\_x) const
- `const_local_iterator` `begin` (`size_type` \_\_n) const
- `const_local_iterator` `cbegin` (`size_type` \_\_n) const
- `const_local_iterator` `end` (`size_type` \_\_n) const
- `const_local_iterator` `cend` (`size_type` \_\_n) const

#### Friends

- template<typename `_Key1` , typename `_Tp1` , typename `_Hash1` , typename `_Pred1` , typename `_Alloc1` > bool **operator==** (const `unordered_multimap`< `_Key1`, `_Tp1`, `_Hash1`, `_Pred1`, `_Alloc1` > &, const `unordered_multimap`< `_Key1`, `_Tp1`, `_Hash1`, `_Pred1`, `_Alloc1` > &)

#### 4.711.1 Detailed Description

```
template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> class std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >
```

A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.

## Template Parameters

<code>_Key</code>	Type of key objects.
<code>_Tp</code>	Type of mapped objects.
<code>_Hash</code>	Hashing function object type, defaults to <code>hash&lt;_Value&gt;</code> .
<code>_Pred</code>	Predicate function object type, defaults to <code>equal_to&lt;_Value&gt;</code> .
<code>_Alloc</code>	Allocator type, defaults to <code>allocator&lt;_Key&gt;</code> .

Meets the requirements of a [container](#), and [unordered associative container](#)

The resulting value type of the container is `std::pair<const _Key, _Tp>`.

Base is `_Hashtable`, dispatched at compile time via template alias `__ummap_hashtable`.

Definition at line 778 of file `unordered_map.h`.

## 4.711.2 Member Typedef Documentation

4.711.2.1 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::allocator_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::allocator_type`

Public typedefs.

Definition at line 792 of file `unordered_map.h`.

4.711.2.2 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::const_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::const_iterator`

Iterator-related typedefs.

Definition at line 802 of file `unordered_map.h`.

4.711.2.3 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::const_local_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::const_local_iterator`

Iterator-related typedefs.

Definition at line 804 of file `unordered_map.h`.

4.711.2.4 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef allocator_type::const_pointer std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::const_pointer`

Iterator-related typedefs.

Definition at line 798 of file `unordered_map.h`.

4.711.2.5 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef allocator_type::const_reference std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::const_reference`

Iterator-related typedefs.

Definition at line 800 of file `unordered_map.h`.

4.711.2.6 `template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::difference_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::difference_type`

Iterator-related typedefs.

Definition at line 806 of file `unordered_map.h`.

4.711.2.7 `template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::hasher std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::hasher`

Public typedefs.

Definition at line 790 of file `unordered_map.h`.

4.711.2.8 `template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::iterator`

Iterator-related typedefs.

Definition at line 801 of file `unordered_map.h`.

4.711.2.9 `template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::key_equal std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::key_equal`

Public typedefs.

Definition at line 791 of file `unordered_map.h`.

4.711.2.10 `template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::key_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::key_type`

Public typedefs.

Definition at line 787 of file `unordered_map.h`.

4.711.2.11 `template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::local_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::local_iterator`

Iterator-related typedefs.

Definition at line 803 of file `unordered_map.h`.

4.711.2.12 `template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::mapped_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::mapped_type`

Public typedefs.

Definition at line 789 of file `unordered_map.h`.

4.711.2.13 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef allocator_type::pointer std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::pointer`

Iterator-related typedefs.

Definition at line 797 of file unordered\_map.h.

4.711.2.14 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef allocator_type::reference std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::reference`

Iterator-related typedefs.

Definition at line 799 of file unordered\_map.h.

4.711.2.15 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::size_type`

Iterator-related typedefs.

Definition at line 805 of file unordered\_map.h.

4.711.2.16 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::value_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::value_type`

Public typedefs.

Definition at line 788 of file unordered\_map.h.

#### 4.711.3 Constructor & Destructor Documentation

4.711.3.1 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap ( size_type __n=10, const hasher & __hf=hasher(), const key_equal & __eq= key_equal(), const allocator_type & __a=allocator_type() ) [inline],[explicit]`

Default constructor creates no elements.

Parameters

<code>__n</code>	Initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eq</code>	A key equality functor.
<code>__a</code>	An allocator object.

Definition at line 819 of file unordered\_map.h.

4.711.3.2 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> template<typename _InputIterator> std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap ( _InputIterator __f, _InputIterator __l, size_type __n=0, const hasher & __hf=hasher(), const key_equal & __eq= key_equal(), const allocator_type & __a= allocator_type() ) [inline]`

Builds an unordered\_multimap from a range.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__n</code>	Minimal initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eqf</code>	A key equality functor.
<code>__a</code>	An allocator object.

Create an `unordered_multimap` consisting of copies of the elements from `[__first,__last)`. This is linear in `N` (where `N` is `distance(__first,__last)`).

Definition at line 840 of file `unordered_map.h`.

```
4.711.3.3  template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc
>::unordered_multimap ( const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > & ) [default]
```

Copy constructor.

```
4.711.3.4  template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc
>::unordered_multimap ( unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > && ) [default]
```

Move constructor.

```
4.711.3.5  template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc
>::unordered_multimap ( initializer_list< value_type > __l, size_type __n = 0, const hasher & __hf
= hasher(), const key_equal & __eqf = key_equal(), const allocator_type & __a = allocator_type() )
[inline]
```

Builds an `unordered_multimap` from an `initializer_list`.

## Parameters

<code>__l</code>	An <code>initializer_list</code> .
<code>__n</code>	Minimal initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eqf</code>	A key equality functor.
<code>__a</code>	An allocator object.

Create an `unordered_multimap` consisting of copies of the elements in the list. This is linear in `N` (where `N` is `__l.size()`).

Definition at line 865 of file `unordered_map.h`.

## 4.711.4 Member Function Documentation

```
4.711.4.1  template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred,
_Alloc >::begin ( ) [inline], [noexcept]
```

Returns a read/write iterator that points to the first element in the `unordered_multimap`.

Definition at line 929 of file `unordered_map.h`.

4.711.4.2 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::begin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the unordered\_multimap.

Definition at line 938 of file unordered\_map.h.

4.711.4.3 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> local_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::begin ( size_type __n ) [inline]`

Returns a read/write iterator pointing to the first bucket element.

Parameters

<code>__n</code>	The bucket index.
------------------	-------------------

Returns

A read/write local iterator.

Definition at line 1277 of file unordered\_map.h.

4.711.4.4 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_local_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::begin ( size_type __n ) const [inline]`

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

<code>__n</code>	The bucket index.
------------------	-------------------

Returns

A read-only local iterator.

Definition at line 1288 of file unordered\_map.h.

4.711.4.5 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::bucket_count ( ) const [inline], [noexcept]`

Returns the number of buckets of the unordered\_multimap.

Definition at line 1244 of file unordered\_map.h.

4.711.4.6 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::cbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the unordered\_multimap.

Definition at line 942 of file unordered\_map.h.

4.711.4.7 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =  
std::allocator<std::pair<const _Key, _Tp>>> const_local_iterator std::unordered_multimap<_Key, _Tp,  
_Hash, _Pred, _Alloc>::cbegin ( size_type __n ) const [inline]`

Returns a read-only (constant) iterator pointing to the first bucket element.

## Parameters

<code>__n</code>	The bucket index.
------------------	-------------------

## Returns

A read-only local iterator.

Definition at line 1292 of file unordered\_map.h.

4.711.4.8 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::cend( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the unordered\_multimap.

Definition at line 964 of file unordered\_map.h.

4.711.4.9 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_local_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::cend( size_type __n ) const [inline]`

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

## Parameters

<code>__n</code>	The bucket index.
------------------	-------------------

## Returns

A read-only local iterator.

Definition at line 1318 of file unordered\_map.h.

4.711.4.10 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::clear( ) [inline], [noexcept]`

Erases all elements in an unordered\_multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1161 of file unordered\_map.h.

4.711.4.11 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::count( const key_type &__x ) const [inline]`

Finds the number of elements.

## Parameters

<code>__x</code>	Key to count.
------------------	---------------

## Returns

Number of elements with specified key.

Definition at line 1221 of file unordered\_map.h.



```
4.711.4.12  template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>,  
              class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename... _Args> iterator  
              std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::emplace ( _Args &&... _args ) [inline]
```

Attempts to build and insert a `std::pair` into the `unordered_multimap`.

## Parameters

<code>__args</code>	Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor).
---------------------	--

## Returns

An iterator that points to the inserted pair.

This function attempts to build and insert a (key, value) pair into the `unordered_multimap`.

Insertion requires amortized constant time.

Definition at line 987 of file `unordered_map.h`.

```
4.711.4.13 template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>,
class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> template<typename... _Args> iterator
std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::emplace_hint ( const_iterator __pos, _Args
&&... __args ) [inline]
```

Attempts to build and insert a `std::pair` into the `unordered_multimap`.

## Parameters

<code>__pos</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__args</code>	Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor).

## Returns

An iterator that points to the element with key of the `std::pair` built from `__args`.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 1013 of file `unordered_map.h`.

```
4.711.4.14 template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> bool std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc
>::empty ( ) const [inline],[noexcept]
```

Returns true if the `unordered_multimap` is empty.

Definition at line 909 of file `unordered_map.h`.

```
4.711.4.15 template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred,
_Alloc>::end ( ) [inline],[noexcept]
```

Returns a read/write iterator that points one past the last element in the `unordered_multimap`.

Definition at line 951 of file `unordered_map.h`.

4.711.4.16 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::end( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the `unordered_multimap`.

Definition at line 960 of file `unordered_map.h`.

4.711.4.17 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> local_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::end( size_type __n ) [inline]`

Returns a read/write iterator pointing to one past the last bucket elements.

Parameters

<code>__n</code>	The bucket index.
------------------	-------------------

Returns

A read/write local iterator.

Definition at line 1303 of file `unordered_map.h`.

4.711.4.18 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_local_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::end( size_type __n ) const [inline]`

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

<code>__n</code>	The bucket index.
------------------	-------------------

Returns

A read-only local iterator.

Definition at line 1314 of file `unordered_map.h`.

4.711.4.19 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::pair<iterator, iterator> std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::equal_range( const key_type & __x ) [inline]`

Finds a subsequence matching given key.

Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

Returns

Pair of iterators that possibly points to the subsequence matching given key.

Definition at line 1232 of file `unordered_map.h`.

4.711.4.20 `template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class  
_Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::pair<const_iterator, const_iterator>  
std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::equal_range ( const key_type & __x ) const  
[inline]`

Finds a subsequence matching given key.

## Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

## Returns

Pair of iterators that possibly points to the subsequence matching given key.

Definition at line 1236 of file `unordered_map.h`.

```
4.711.4.21  template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred,
            _Alloc>::erase ( const_iterator __position ) [inline]
```

Erases an element from an `unordered_multimap`.

## Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

## Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an `unordered_multimap`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1111 of file `unordered_map.h`.

```
4.711.4.22  template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred,
            _Alloc>::erase ( iterator __it ) [inline]
```

Erases an element from an `unordered_multimap`.

## Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

## Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an `unordered_multimap`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1116 of file `unordered_map.h`.

```
4.711.4.23  template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred,
            _Alloc>::erase ( const key_type & __x ) [inline]
```

Erases elements according to the provided key.

## Parameters

<code>__x</code>	Key of elements to be erased.
------------------	-------------------------------

## Returns

The number of elements erased.

This function erases all the elements located by the given key from an `unordered_multimap`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1132 of file `unordered_map.h`.

```
4.711.4.24 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred,
_Alloc >::erase ( const_iterator __first, const_iterator __last ) [inline]
```

Erases a [`__first`,`__last`) range of elements from an `unordered_multimap`.

## Parameters

<code>__first</code>	Iterator pointing to the start of the range to be erased.
<code>__last</code>	Iterator pointing to the end of the range to be erased.

## Returns

The iterator `__last`.

This function erases a sequence of elements from an `unordered_multimap`. Note that this function only erases the elements, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1151 of file `unordered_map.h`.

```
4.711.4.25 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred,
_Alloc >::find ( const key_type & __x ) [inline]
```

Tries to locate an element in an `unordered_multimap`.

## Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

## Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 1207 of file `unordered_map.h`.

```
4.711.4.26 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::unordered_multimap< _Key, _Tp, _Hash,
_Pred, _Alloc >::find ( const key_type & __x ) const [inline]
```

Tries to locate an element in an `unordered_multimap`.

## Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

## Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 1211 of file `unordered_map.h`.

```
4.711.4.27 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> allocator_type std::unordered_multimap<_Key, _Tp, _Hash,
_Pred, _Alloc>::get_allocator ( ) const [inline], [noexcept]
```

Returns the allocator object with which the `unordered_multimap` was constructed.

Definition at line 902 of file `unordered_map.h`.

```
4.711.4.28 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> hasher std::unordered_multimap<_Key, _Tp, _Hash, _Pred,
_Alloc>::hash_function ( ) const [inline]
```

Returns the hash functor object with which the `unordered_multimap` was constructed.

Definition at line 1183 of file `unordered_map.h`.

```
4.711.4.29 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred,
_Alloc>::insert ( const value_type & __x ) [inline]
```

Inserts a `std::pair` into the `unordered_multimap`.

## Parameters

<code>__x</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).
------------------	---

## Returns

An iterator that points to the inserted pair.

Insertion requires amortized constant time.

Definition at line 1027 of file `unordered_map.h`.

```
4.711.4.30 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class
_Alloc = std::allocator<std::pair<const _Key, _Tp>>> template<typename _Pair, typename = typename
std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type> iterator std::unordered_multimap<
_Key, _Tp, _Hash, _Pred, _Alloc>::insert ( _Pair && __x ) [inline]
```

Inserts a `std::pair` into the `unordered_multimap`.

## Parameters

<code>__x</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).
------------------	---

## Returns

An iterator that points to the inserted pair.

Insertion requires amortized constant time.

Definition at line 1034 of file `unordered_map.h`.

```
4.711.4.31 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred,
_Alloc >::insert( const_iterator __hint, const value_type & __x ) [inline]
```

Inserts a `std::pair` into the `unordered_multimap`.

## Parameters

<code>__hint</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__x</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).

## Returns

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 1059 of file `unordered_map.h`.

```
4.711.4.32 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class
_Alloc = std::allocator<std::pair<const _Key, _Tp>>> template<typename _Pair, typename = typename
std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type> iterator std::unordered_multimap<
_Key, _Tp, _Hash, _Pred, _Alloc >::insert( const_iterator __hint, _Pair && __x ) [inline]
```

Inserts a `std::pair` into the `unordered_multimap`.

## Parameters

<code>__hint</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__x</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).

## Returns

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 1066 of file `unordered_map.h`.



4.711.4.33 `template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename _InputIterator > void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert ( _InputIterator __first, _InputIterator __last ) [inline]`

A template function that attempts to insert a range of elements.

Parameters

<code>__first</code>	Iterator pointing to the start of the range to be inserted.
<code>__last</code>	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 1081 of file `unordered_map.h`.

4.711.4.34 `template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert ( initializer_list< value_type > __l ) [inline]`

Attempts to insert a list of elements into the `unordered_multimap`.

Parameters

<code>__l</code>	A <code>std::initializer_list&lt;value_type&gt;</code> of elements to be inserted.
------------------	--

Complexity similar to that of the range constructor.

Definition at line 1093 of file `unordered_map.h`.

4.711.4.35 `template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> key_equal std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::key_eq ( ) const [inline]`

Returns the key comparison object with which the `unordered_multimap` was constructed.

Definition at line 1189 of file `unordered_map.h`.

4.711.4.36 `template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> float std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::load_factor ( ) const [inline], [noexcept]`

Returns the average number of elements per bucket.

Definition at line 1326 of file `unordered_map.h`.

4.711.4.37 `template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::max_bucket_count ( ) const [inline], [noexcept]`

Returns the maximum number of buckets of the `unordered_multimap`.

Definition at line 1249 of file `unordered_map.h`.

4.711.4.38 `template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> float std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor ( ) const [inline], [noexcept]`

Returns a positive number that the `unordered_multimap` tries to keep the load factor less than or equal to.

Definition at line 1332 of file `unordered_map.h`.

```
4.711.4.39 template<class _Key , class _Tp , class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =  
std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc  
>::max_load_factor( float __z ) [inline]
```

Change the unordered\_multimap maximum load factor.

## Parameters

<code>__z</code>	The new maximum load factor.
------------------	------------------------------

Definition at line 1340 of file `unordered_map.h`.

```
4.711.4.40 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred,
_Alloc>::max_size( ) const [inline], [noexcept]
```

Returns the maximum size of the `unordered_multimap`.

Definition at line 919 of file `unordered_map.h`.

```
4.711.4.41 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> unordered_multimap& std::unordered_multimap<_Key,
_Tp, _Hash, _Pred, _Alloc>::operator=( const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> & )
[default]
```

Copy assignment operator.

```
4.711.4.42 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> unordered_multimap& std::unordered_multimap<_Key,
_Tp, _Hash, _Pred, _Alloc>::operator=( unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> && )
[default]
```

Move assignment operator.

```
4.711.4.43 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> unordered_multimap& std::unordered_multimap<_Key, _Tp,
_Hash, _Pred, _Alloc>::operator=( initializer_list<value_type> __l ) [inline]
```

`Unordered_multimap` list assignment operator.

## Parameters

<code>__l</code>	An <code>initializer_list</code> .
------------------	------------------------------------

This function fills an `unordered_multimap` with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the `unordered_multimap` and that the resulting `unordered_multimap`'s size is the same as the number of elements assigned. Old data may be lost.

Definition at line 893 of file `unordered_map.h`.

```
4.711.4.44 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>
::rehash( size_type __n ) [inline]
```

May rehash the `unordered_multimap`.

## Parameters

<code>__n</code>	The new number of buckets.
------------------	----------------------------

Rehash will occur only if the new number of buckets respect the `unordered_multimap` maximum load factor.

Definition at line 1351 of file `unordered_map.h`.

```
4.711.4.45 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =  
std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc  
>::reserve ( size_type __n ) [inline]
```

Prepare the unordered\_multimap for a specified number of elements.

## Parameters

<code>__n</code>	Number of elements required.
------------------	------------------------------

Same as `rehash(ceil(n / max_load_factor()))`.

Definition at line 1362 of file `unordered_map.h`.

4.711.4.46 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::size ( ) const [inline], [noexcept]`

Returns the size of the `unordered_multimap`.

Definition at line 914 of file `unordered_map.h`.

4.711.4.47 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::swap ( unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__x ) [inline]`

Swaps data with another `unordered_multimap`.

## Parameters

<code>__x</code>	An <code>unordered_multimap</code> of the same element and allocator types.
------------------	---

This exchanges the elements between two `unordered_multimap` in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

Definition at line 1175 of file `unordered_map.h`.

The documentation for this class was generated from the following file:

- [unordered\\_map.h](#)

## 4.712 `std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>` Class Template Reference

Inherits `std::__allow_copy_cons<bool>`.

## Public Types

- `typedef _Hashtable::key_type` [key\\_type](#)
- `typedef _Hashtable::value_type` [value\\_type](#)
- `typedef _Hashtable::hasher` [hasher](#)
- `typedef _Hashtable::key_equal` [key\\_equal](#)
- `typedef _Hashtable::allocator_type` [allocator\\_type](#)
- `typedef allocator_type::pointer` [pointer](#)
- `typedef allocator_type::const_pointer` [const\\_pointer](#)
- `typedef allocator_type::reference` [reference](#)
- `typedef allocator_type::const_reference` [const\\_reference](#)
- `typedef _Hashtable::iterator` [iterator](#)
- `typedef _Hashtable::const_iterator` [const\\_iterator](#)
- `typedef _Hashtable::local_iterator` [local\\_iterator](#)
- `typedef _Hashtable::const_local_iterator` [const\\_local\\_iterator](#)
- `typedef _Hashtable::size_type` [size\\_type](#)
- `typedef _Hashtable::difference_type` [difference\\_type](#)

## Public Member Functions

- `unordered_multiset` (`size_type` \_\_n=10, const `hasher` &\_\_hf=`hasher`(), const `key_equal` &\_\_eq=`key_equal`(), const `allocator_type` &\_\_a=`allocator_type`())
- `template<typename _InputIterator > unordered_multiset` (`_InputIterator` \_\_f, `_InputIterator` \_\_l, `size_type` \_\_n=0, const `hasher` &\_\_hf=`hasher`(), const `key_equal` &\_\_eq=`key_equal`(), const `allocator_type` &\_\_a=`allocator_type`())
- `unordered_multiset` (const `unordered_multiset` &)=default
- `unordered_multiset` (`unordered_multiset` &&)=default
- `unordered_multiset` (`initializer_list`< `value_type` > \_\_l, `size_type` \_\_n=0, const `hasher` &\_\_hf=`hasher`(), const `key_equal` &\_\_eq=`key_equal`(), const `allocator_type` &\_\_a=`allocator_type`())
- `size_type bucket` (const `key_type` &\_\_key) const
- `size_type bucket_count` () const noexcept
- `size_type bucket_size` (`size_type` \_\_n) const
- `const_iterator cbegin` () const noexcept
- `const_iterator cend` () const noexcept
- `void clear` () noexcept
- `size_type count` (const `key_type` &\_\_x) const
- `template<typename... _Args> iterator emplace` (`_Args` &&...\_\_args)
- `template<typename... _Args> iterator emplace_hint` (const `iterator` \_\_pos, `_Args` &&...\_\_args)
- `bool empty` () const noexcept
- `size_type erase` (const `key_type` &\_\_x)
- `iterator erase` (const `iterator` \_\_first, const `iterator` \_\_last)
- `allocator_type get_allocator` () const noexcept
- `hasher hash_function` () const
- `template<typename _InputIterator > void insert` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `void insert` (`initializer_list`< `value_type` > \_\_l)
- `key_equal key_eq` () const
- `float load_factor` () const noexcept
- `size_type max_bucket_count` () const noexcept
- `float max_load_factor` () const noexcept
- `void max_load_factor` (`float` \_\_z)
- `size_type max_size` () const noexcept
- `unordered_multiset & operator=` (const `unordered_multiset` &)=default
- `unordered_multiset & operator=` (`unordered_multiset` &&\_\_x)=default
- `unordered_multiset & operator=` (`initializer_list`< `value_type` > \_\_l)
- `void rehash` (`size_type` \_\_n)
- `void reserve` (`size_type` \_\_n)
- `size_type size` () const noexcept
- `void swap` (`unordered_multiset` &\_\_x)
  
- `iterator begin` () noexcept
- `const_iterator begin` () const noexcept
  
- `iterator end` () noexcept
- `const_iterator end` () const noexcept
  
- `iterator insert` (const `value_type` &\_\_x)
- `iterator insert` (`value_type` &&\_\_x)
  
- `iterator insert` (const `iterator` \_\_hint, const `value_type` &\_\_x)
- `iterator insert` (const `iterator` \_\_hint, `value_type` &&\_\_x)

- `iterator erase (const_iterator __position)`
- `iterator erase (iterator __it)`
- `iterator find (const key_type &__x)`
- `const_iterator find (const key_type &__x) const`
- `std::pair< iterator, iterator > equal_range (const key_type &__x)`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const`
- `local_iterator begin (size_type __n)`
- `const_local_iterator begin (size_type __n) const`
- `const_local_iterator cbegin (size_type __n) const`
- `local_iterator end (size_type __n)`
- `const_local_iterator end (size_type __n) const`
- `const_local_iterator cend (size_type __n) const`

#### Friends

- `template<typename _Value1, typename _Hash1, typename _Pred1, typename _Alloc1 > bool operator== (const unordered_multiset<_Value1, _Hash1, _Pred1, _Alloc1 > &, const unordered_multiset<_Value1, _Hash1, _Pred1, _Alloc1 > &)`

#### 4.712.1 Detailed Description

`template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> class std::unordered_multiset<_Value, _Hash, _Pred, _Alloc >`

A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.

##### Template Parameters

<code>_Value</code>	Type of key objects.
<code>_Hash</code>	Hashing function object type, defaults to <code>hash&lt;_Value&gt;</code> .
<code>_Pred</code>	Predicate function object type, defaults to <code>equal_to&lt;_Value&gt;</code> .
<code>_Alloc</code>	Allocator type, defaults to <code>allocator&lt;_Key&gt;</code> .

Meets the requirements of a `container`, and `unordered associative container`

Base is `_Hashtable`, dispatched at compile time via template alias `__umset_hashtable`.

Definition at line 698 of file `unordered_set.h`.

#### 4.712.2 Member Typedef Documentation

4.712.2.1 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::allocator_type std::unordered_multiset<_Value, _Hash, _Pred, _Alloc >::allocator_type`

Public typedefs.

Definition at line 711 of file `unordered_set.h`.

4.712.2.2 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::const_iterator`

Iterator-related typedefs.

Definition at line 721 of file unordered\_set.h.

4.712.2.3 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::const_local_iterator`

Iterator-related typedefs.

Definition at line 723 of file unordered\_set.h.

4.712.2.4 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef allocator_type::const_pointer std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::const_pointer`

Iterator-related typedefs.

Definition at line 717 of file unordered\_set.h.

4.712.2.5 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef allocator_type::const_reference std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::const_reference`

Iterator-related typedefs.

Definition at line 719 of file unordered\_set.h.

4.712.2.6 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::difference_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::difference_type`

Iterator-related typedefs.

Definition at line 725 of file unordered\_set.h.

4.712.2.7 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::hasher std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::hasher`

Public typedefs.

Definition at line 709 of file unordered\_set.h.

4.712.2.8 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::iterator`

Iterator-related typedefs.

Definition at line 720 of file unordered\_set.h.



4.712.2.9 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::key_equal std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::key_equal`

Public typedefs.

Definition at line 710 of file `unordered_set.h`.

4.712.2.10 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::key_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::key_type`

Public typedefs.

Definition at line 707 of file `unordered_set.h`.

4.712.2.11 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::local_iterator`

Iterator-related typedefs.

Definition at line 722 of file `unordered_set.h`.

4.712.2.12 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef allocator_type::pointer std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::pointer`

Iterator-related typedefs.

Definition at line 716 of file `unordered_set.h`.

4.712.2.13 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef allocator_type::reference std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::reference`

Iterator-related typedefs.

Definition at line 718 of file `unordered_set.h`.

4.712.2.14 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::size_type`

Iterator-related typedefs.

Definition at line 724 of file `unordered_set.h`.

4.712.2.15 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::value_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::value_type`

Public typedefs.

Definition at line 708 of file `unordered_set.h`.

#### 4.712.3 Constructor & Destructor Documentation

```
4.712.3.1 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =  
std::allocator<_Value>> std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::unordered_multiset  
( size_type __n = 10, const hasher & __hf = hasher(), const key_equal & __eq/ = key_equal(), const  
allocator_type & __a = allocator_type() ) [inline], [explicit]
```

Default constructor creates no elements.

## Parameters

<code>__n</code>	Initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eq/</code>	A key equality functor.
<code>__a</code>	An allocator object.

Definition at line 737 of file `unordered_set.h`.

```
4.712.3.2  template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> template<typename _InputIterator > std::unordered_multiset< _Value, _Hash, _Pred,
_Alloc >::unordered_multiset ( _InputIterator __f, _InputIterator __l, size_type __n = 0, const hasher & __hf
= hasher(), const key_equal & __eq/ = key_equal(), const allocator_type & __a = allocator_type() )
[inline]
```

Builds an `unordered_multiset` from a range.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__n</code>	Minimal initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eq/</code>	A key equality functor.
<code>__a</code>	An allocator object.

Create an `unordered_multiset` consisting of copies of the elements from `[__first,__last)`. This is linear in `N` (where `N` is `distance(__first,__last)`).

Definition at line 758 of file `unordered_set.h`.

```
4.712.3.3  template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
const unordered_multiset< _Value, _Hash, _Pred, _Alloc > & ) [default]
```

Copy constructor.

```
4.712.3.4  template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
unordered_multiset< _Value, _Hash, _Pred, _Alloc > && ) [default]
```

Move constructor.

```
4.712.3.5  template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
initializer_list< value_type > __l, size_type __n = 0, const hasher & __hf = hasher(), const key_equal & __eq/
= key_equal(), const allocator_type & __a = allocator_type() ) [inline]
```

Builds an `unordered_multiset` from an `initializer_list`.

## Parameters

<code>__l</code>	An <code>initializer_list</code> .
<code>__n</code>	Minimal initial number of buckets.

<code>__hf</code>	A hash functor.
<code>__eq/</code>	A key equality functor.
<code>__a</code>	An allocator object.

Create an `unordered_multiset` consisting of copies of the elements in the list. This is linear in N (where N is `__l.size()`).

Definition at line 783 of file `unordered_set.h`.

#### 4.712.4 Member Function Documentation

**4.712.4.1** `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc >::begin ( )`  
`[inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the `unordered_multiset`.

Definition at line 848 of file `unordered_set.h`.

**4.712.4.2** `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc >::begin ( )`  
`const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the `unordered_multiset`.

Definition at line 852 of file `unordered_set.h`.

**4.712.4.3** `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> local_iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc >::begin (`  
`size_type __n ) [inline]`

Returns a read-only (constant) iterator pointing to the first bucket element.

##### Parameters

<code>__n</code>	The bucket index.
------------------	-------------------

##### Returns

A read-only local iterator.

Definition at line 1175 of file `unordered_set.h`.

**4.712.4.4** `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_local_iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc >::begin`  
`( size_type __n ) const [inline]`

Returns a read-only (constant) iterator pointing to the first bucket element.

##### Parameters

<code>__n</code>	The bucket index.
------------------	-------------------

##### Returns

A read-only local iterator.

Definition at line 1179 of file `unordered_set.h`.

4.712.4.5 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> size_type std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::bucket_count ( ) const [inline], [noexcept]`

Returns the number of buckets of the unordered\_multiset.

Definition at line 1141 of file unordered\_set.h.

4.712.4.6 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::cbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the unordered\_multiset.

Definition at line 875 of file unordered\_set.h.

4.712.4.7 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_local_iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::cbegin ( size_type __n ) const [inline]`

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

<code>__n</code>	The bucket index.
------------------	-------------------

Returns

A read-only local iterator.

Definition at line 1183 of file unordered\_set.h.

4.712.4.8 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::cend ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the unordered\_multiset.

Definition at line 883 of file unordered\_set.h.

4.712.4.9 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_local_iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::cend ( size_type __n ) const [inline]`

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

<code>__n</code>	The bucket index.
------------------	-------------------

**Returns**

A read-only local iterator.

Definition at line 1203 of file unordered\_set.h.

```
4.712.4.10 template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
           = std::allocator<_Value>> void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::clear ( )
           [inline], [noexcept]
```

Erases all elements in an unordered\_multiset.

Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1059 of file unordered\_set.h.

```
4.712.4.11 template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
           = std::allocator<_Value>> size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::count ( const
           key_type & __x ) const [inline]
```

Finds the number of elements.

**Parameters**

<code>__x</code>	Element to located.
------------------	---------------------

**Returns**

Number of elements with specified key.

Definition at line 1118 of file unordered\_set.h.

```
4.712.4.12 template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
           = std::allocator<_Value>> template<typename... _Args> iterator std::unordered_multiset< _Value, _Hash,
           _Pred, _Alloc >::emplace ( _Args &&... __args ) [inline]
```

Builds and insert an element into the unordered\_multiset.

**Parameters**

<code>__args</code>	Arguments used to generate an element.
---------------------	--

**Returns**

An iterator that points to the inserted element.

Insertion requires amortized constant time.

Definition at line 897 of file unordered\_set.h.

```
4.712.4.13 template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
           = std::allocator<_Value>> template<typename... _Args> iterator std::unordered_multiset< _Value, _Hash,
           _Pred, _Alloc >::emplace_hint ( const_iterator __pos, _Args &&... __args ) [inline]
```

Inserts an element into the unordered\_multiset.

## Parameters

<code>__pos</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__args</code>	Arguments used to generate the element to be inserted.

## Returns

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html>.

Insertion requires amortized constant time.

Definition at line 919 of file `unordered_set.h`.

```
4.712.4.14  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> bool std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::empty ( ) const
            [inline], [noexcept]
```

Returns true if the `unordered_multiset` is empty.

Definition at line 827 of file `unordered_set.h`.

```
4.712.4.15  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::end ( )
            [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the `unordered_multiset`.

Definition at line 862 of file `unordered_set.h`.

```
4.712.4.16  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> const_iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::end ( )
            const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the `unordered_multiset`.

Definition at line 866 of file `unordered_set.h`.

```
4.712.4.17  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> local_iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::end (
            size_type __n ) [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

## Parameters

<code>__n</code>	The bucket index.
------------------	-------------------

## Returns

A read-only local iterator.

Definition at line 1195 of file `unordered_set.h`.

```
4.712.4.18 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =  
std::allocator<_Value>> const_local_iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc >::end  
( size_type __n )const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.



## Parameters

<code>__n</code>	The bucket index.
------------------	-------------------

## Returns

A read-only local iterator.

Definition at line 1199 of file `unordered_set.h`.

```
4.712.4.19  template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> std::pair<iterator, iterator> std::unordered_multiset< _Value, _Hash, _Pred,
            _Alloc >::equal_range ( const key_type & __x )  [inline]
```

Finds a subsequence matching given key.

## Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

## Returns

Pair of iterators that possibly points to the subsequence matching given key.

Definition at line 1129 of file `unordered_set.h`.

```
4.712.4.20  template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> std::pair<const_iterator, const_iterator> std::unordered_multiset< _Value,
            _Hash, _Pred, _Alloc >::equal_range ( const key_type & __x ) const  [inline]
```

Finds a subsequence matching given key.

## Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

## Returns

Pair of iterators that possibly points to the subsequence matching given key.

Definition at line 1133 of file `unordered_set.h`.

```
4.712.4.21  template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (
            const_iterator __position )  [inline]
```

Erases an element from an `unordered_multiset`.

## Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

## Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an `unordered_multiset`.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1005 of file unordered\_set.h.

```
4.712.4.22 template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase ( iterator
__it ) [inline]
```

Erases an element from an unordered\_multiset.

Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered\_multiset.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1010 of file unordered\_set.h.

```
4.712.4.23 template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase ( const
key_type & __x ) [inline]
```

Erases elements according to the provided key.

Parameters

<code>__x</code>	Key of element to be erased.
------------------	------------------------------

Returns

The number of elements erased.

This function erases all the elements located by the given key from an unordered\_multiset.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1028 of file unordered\_set.h.

```
4.712.4.24 template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (
const_iterator __first, const_iterator __last ) [inline]
```

Erases a [`__first`,`__last`) range of elements from an unordered\_multiset.

Parameters

---

<code>__first</code>	Iterator pointing to the start of the range to be erased.
<code>__last</code>	Iterator pointing to the end of the range to be erased.

**Returns**

The iterator `__last`.

This function erases a sequence of elements from an `unordered_multiset`.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1048 of file `unordered_set.h`.

```
4.712.4.25  template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::find ( const
            key_type & __x ) [inline]
```

Tries to locate an element in an `unordered_multiset`.

**Parameters**

<code>__x</code>	Element to be located.
------------------	------------------------

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 1104 of file `unordered_set.h`.

```
4.712.4.26  template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> const_iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::find ( const
            key_type & __x ) const [inline]
```

Tries to locate an element in an `unordered_multiset`.

**Parameters**

<code>__x</code>	Element to be located.
------------------	------------------------

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 1108 of file `unordered_set.h`.

```
4.712.4.27  template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc
            = std::allocator<_Value>> allocator_type std::unordered_multiset<_Value, _Hash, _Pred, _Alloc
            >::get_allocator ( ) const [inline], [noexcept]
```

Returns the allocator object with which the `unordered_multiset` was constructed.

Definition at line 820 of file `unordered_set.h`.

4.712.4.28 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> hasher std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::hash_function ( ) const [inline]`

Returns the hash functor object with which the unordered\_multiset was constructed.

Definition at line 1080 of file unordered\_set.h.

4.712.4.29 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert ( const value_type & __x ) [inline]`

Inserts an element into the unordered\_multiset.

Parameters

<code>__x</code>	Element to be inserted.
------------------	-------------------------

Returns

An iterator that points to the inserted element.

Insertion requires amortized constant time.

Definition at line 931 of file unordered\_set.h.

4.712.4.30 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert ( value_type && __x ) [inline]`

Inserts an element into the unordered\_multiset.

Parameters

<code>__x</code>	Element to be inserted.
------------------	-------------------------

Returns

An iterator that points to the inserted element.

Insertion requires amortized constant time.

Definition at line 935 of file unordered\_set.h.

References std::move().

4.712.4.31 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert ( const_iterator __hint, const value_type & __x ) [inline]`

Inserts an element into the unordered\_multiset.

Parameters

<code>__hint</code>	An iterator that serves as a hint as to where the element should be inserted.
---------------------	---

<code>__x</code>	Element to be inserted.
------------------	-------------------------

**Returns**

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.↵html>

Insertion requires amortized constant.

Definition at line 957 of file `unordered_set.h`.

```
4.712.4.32 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::insert (
const_iterator __hint, value_type && __x ) [inline]
```

Inserts an element into the `unordered_multiset`.

**Parameters**

<code>__hint</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__x</code>	Element to be inserted.

**Returns**

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.↵html>

Insertion requires amortized constant.

Definition at line 961 of file `unordered_set.h`.

References `std::move()`.

```
4.712.4.33 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> template<typename _InputIterator> void std::unordered_multiset<_Value, _Hash,
_Pred, _Alloc>::insert ( _InputIterator __first, _InputIterator __last ) [inline]
```

A template function that inserts a range of elements.

**Parameters**

<code>__first</code>	Iterator pointing to the start of the range to be inserted.
<code>__last</code>	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 975 of file `unordered_set.h`.

```
4.712.4.34  template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =  
            std::allocator<_Value>> void std::unordered_multiset<_Value, _Hash, _Pred, _Alloc >::insert ( initializer_list<  
            value_type > __l ) [inline]
```

Inserts a list of elements into the unordered\_multiset.

## Parameters

<code>__l</code>	A <code>std::initializer_list&lt;value_type&gt;</code> of elements to be inserted.
------------------	--

Complexity similar to that of the range constructor.

Definition at line 986 of file `unordered_set.h`.

```
4.712.4.35 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> key_equal std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::key_eq ( )
const [inline]
```

Returns the key comparison object with which the `unordered_multiset` was constructed.

Definition at line 1086 of file `unordered_set.h`.

```
4.712.4.36 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> float std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::load_factor ( ) const
[inline], [noexcept]
```

Returns the average number of elements per bucket.

Definition at line 1211 of file `unordered_set.h`.

```
4.712.4.37 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> size_type std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::max_bucket_count ( ) const
[inline], [noexcept]
```

Returns the maximum number of buckets of the `unordered_multiset`.

Definition at line 1146 of file `unordered_set.h`.

```
4.712.4.38 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> float std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::max_load_factor ( )
const [inline], [noexcept]
```

Returns a positive number that the `unordered_multiset` tries to keep the load factor less than or equal to.

Definition at line 1217 of file `unordered_set.h`.

```
4.712.4.39 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> void std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::max_load_factor ( float
__z ) [inline]
```

Change the `unordered_multiset` maximum load factor.

## Parameters

<code>__z</code>	The new maximum load factor.
------------------	------------------------------

Definition at line 1225 of file `unordered_set.h`.

```
4.712.4.40 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> size_type std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::max_size ( )
const [inline], [noexcept]
```

Returns the maximum size of the `unordered_multiset`.

Definition at line 837 of file `unordered_set.h`.

4.712.4.41 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> unordered_multiset& std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::operator= ( const unordered_multiset<_Value, _Hash, _Pred, _Alloc> & ) [default]`

Copy assignment operator.

4.712.4.42 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> unordered_multiset& std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::operator= ( unordered_multiset<_Value, _Hash, _Pred, _Alloc> &&_x ) [default]`

Move assignment operator.

4.712.4.43 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> unordered_multiset& std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::operator= ( initializer_list<value_type> >_l ) [inline]`

Unordered\_multiset list assignment operator.

Parameters

<code>_l</code>	An initializer_list.
-----------------	----------------------

This function fills an unordered\_multiset with copies of the elements in the initializer list `_l`.

Note that the assignment completely changes the unordered\_multiset and that the resulting unordered\_set's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 811 of file unordered\_set.h.

4.712.4.44 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::rehash ( size_type __n ) [inline]`

May rehash the unordered\_multiset.

Parameters

<code>__n</code>	The new number of buckets.
------------------	----------------------------

Rehash will occur only if the new number of buckets respect the unordered\_multiset maximum load factor.

Definition at line 1236 of file unordered\_set.h.

4.712.4.45 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::reserve ( size_type __n ) [inline]`

Prepare the unordered\_multiset for a specified number of elements.

Parameters

<code>__n</code>	Number of elements required.
------------------	------------------------------

Same as `rehash(ceil(n / max_load_factor()))`.

Definition at line 1247 of file unordered\_set.h.

4.712.4.46 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> size_type std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::size ( ) const [inline], [noexcept]`

Returns the size of the unordered\_multiset.



Definition at line 832 of file unordered\_set.h.

```
4.712.4.47 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc
           = std::allocator<_Value>> void std::unordered_multiset<_Value, _Hash, _Pred, _Alloc >::swap (
           unordered_multiset<_Value, _Hash, _Pred, _Alloc > &__x ) [inline]
```

Swaps data with another unordered\_multiset.

#### Parameters

__x	An unordered_multiset of the same element and allocator types.
-----	--

This exchanges the elements between two sets in constant time. Note that the global std::swap() function is specialized such that std::swap(s1,s2) will feed to this function.

Definition at line 1072 of file unordered\_set.h.

The documentation for this class was generated from the following file:

- [unordered\\_set.h](#)

### 4.713 std::unordered\_set<\_Value, \_Hash, \_Pred, \_Alloc > Class Template Reference

Inherits std::\_\_allow\_copy\_cons< bool >.

#### Public Types

- typedef \_Hashtable::key\_type [key\\_type](#)
- typedef \_Hashtable::value\_type [value\\_type](#)
- typedef \_Hashtable::hasher [hasher](#)
- typedef \_Hashtable::key\_equal [key\\_equal](#)
- typedef \_Hashtable::allocator\_type [allocator\\_type](#)
- typedef allocator\_type::pointer [pointer](#)
- typedef allocator\_type::const\_pointer [const\\_pointer](#)
- typedef allocator\_type::reference [reference](#)
- typedef allocator\_type::const\_reference [const\\_reference](#)
- typedef \_Hashtable::iterator [iterator](#)
- typedef \_Hashtable::const\_iterator [const\\_iterator](#)
- typedef \_Hashtable::local\_iterator [local\\_iterator](#)
- typedef \_Hashtable::const\_local\_iterator [const\\_local\\_iterator](#)
- typedef \_Hashtable::size\_type [size\\_type](#)
- typedef \_Hashtable::difference\_type [difference\\_type](#)

#### Public Member Functions

- [unordered\\_set](#) ([size\\_type](#) \_\_n=10, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- template<typename \_InputIterator > [unordered\\_set](#) (\_InputIterator \_\_f, \_InputIterator \_\_l, [size\\_type](#) \_\_n=0, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [unordered\\_set](#) (const [unordered\\_set](#) &)=default
- [unordered\\_set](#) ([unordered\\_set](#) &&)=default
- [unordered\\_set](#) (initializer\_list< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n=0, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_↵](#)  
equal &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())

- `size_type bucket` (const `key_type` &\_\_key) const
  - `size_type bucket_count` () const noexcept
  - `size_type bucket_size` (`size_type` \_\_n) const
  - `const_iterator cbegin` () const noexcept
  - `const_iterator cend` () const noexcept
  - void `clear` () noexcept
  - `size_type count` (const `key_type` &\_\_x) const
  - template<typename... \_Args> `std::pair< iterator, bool > emplace` (\_Args &&...\_\_args)
  - template<typename... \_Args> `iterator emplace_hint` (const\_iterator \_\_pos, \_Args &&...\_\_args)
  - bool `empty` () const noexcept
  - `size_type erase` (const `key_type` &\_\_x)
  - `iterator erase` (const\_iterator \_\_first, const\_iterator \_\_last)
  - `allocator_type get_allocator` () const noexcept
  - `hasher hash_function` () const
  - template<typename \_InputIterator > void `insert` (\_InputIterator \_\_first, \_InputIterator \_\_last)
  - void `insert` (initializer\_list< `value_type` > \_\_l)
  - `key_equal key_eq` () const
  - float `load_factor` () const noexcept
  - `size_type max_bucket_count` () const noexcept
  - float `max_load_factor` () const noexcept
  - void `max_load_factor` (float \_\_z)
  - `size_type max_size` () const noexcept
  - `unordered_set & operator=` (const `unordered_set` &)=default
  - `unordered_set & operator=` (`unordered_set` &&)=default
  - `unordered_set & operator=` (initializer\_list< `value_type` > \_\_l)
  - void `rehash` (`size_type` \_\_n)
  - void `reserve` (`size_type` \_\_n)
  - `size_type size` () const noexcept
  - void `swap` (`unordered_set` &\_\_x)
- 
- `iterator begin` () noexcept
  - `const_iterator begin` () const noexcept
- 
- `iterator end` () noexcept
  - `const_iterator end` () const noexcept
- 
- `std::pair< iterator, bool > insert` (const `value_type` &\_\_x)
  - `std::pair< iterator, bool > insert` (`value_type` &&\_\_x)
- 
- `iterator insert` (const\_iterator \_\_hint, const `value_type` &\_\_x)
  - `iterator insert` (const\_iterator \_\_hint, `value_type` &&\_\_x)
- 
- `iterator erase` (const\_iterator \_\_position)
  - `iterator erase` (iterator \_\_it)
- 
- `iterator find` (const `key_type` &\_\_x)
  - `const_iterator find` (const `key_type` &\_\_x) const
- 
- `std::pair< iterator, iterator > equal_range` (const `key_type` &\_\_x)
  - `std::pair< const_iterator, const_iterator > equal_range` (const `key_type` &\_\_x) const

- [local\\_iterator begin \(size\\_type \\_\\_n\)](#)
- [const\\_local\\_iterator begin \(size\\_type \\_\\_n\) const](#)
- [const\\_local\\_iterator cbegin \(size\\_type \\_\\_n\) const](#)
- [local\\_iterator end \(size\\_type \\_\\_n\)](#)
- [const\\_local\\_iterator end \(size\\_type \\_\\_n\) const](#)
- [const\\_local\\_iterator cend \(size\\_type \\_\\_n\) const](#)

#### Friends

- `template<typename _Value1, typename _Hash1, typename _Pred1, typename _Alloc1 > bool operator== (const unordered\_set<_Value1, _Hash1, _Pred1, _Alloc1 > &, const unordered\_set<_Value1, _Hash1, _Pred1, _Alloc1 > &)`

#### 4.713.1 Detailed Description

`template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> class std::unordered_set<_Value, _Hash, _Pred, _Alloc >`

A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.

#### Template Parameters

<code>_Value</code>	Type of key objects.
<code>_Hash</code>	Hashing function object type, defaults to <code>hash&lt;_Value&gt;</code> .
<code>_Pred</code>	Predicate function object type, defaults to <code>equal_to&lt;_Value&gt;</code> .
<code>_Alloc</code>	Allocator type, defaults to <code>allocator&lt;_Key&gt;</code> .

Meets the requirements of a [container](#), and [unordered associative container](#)

Base is `_Hashtable`, dispatched at compile time via template alias `__uset_hashtable`.

Definition at line 93 of file `unordered_set.h`.

#### 4.713.2 Member Typedef Documentation

4.713.2.1 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::allocator_type std::unordered_set<_Value, _Hash, _Pred, _Alloc>::allocator_type`

Public typedefs.

Definition at line 106 of file `unordered_set.h`.

4.713.2.2 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::const_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::const_iterator`

Iterator-related typedefs.

Definition at line 116 of file `unordered_set.h`.

4.713.2.3 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::const_local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::const_local_iterator`

Iterator-related typedefs.

Definition at line 118 of file unordered\_set.h.

```
4.713.2.4 template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =  
std::allocator<_Value>> typedef allocator_type::const_pointer std::unordered_set< _Value, _Hash, _Pred, _Alloc  
>::const_pointer
```

Iterator-related typedefs.

Definition at line 112 of file unordered\_set.h.

```
4.713.2.5 template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =  
std::allocator<_Value>> typedef allocator_type::const_reference std::unordered_set< _Value, _Hash, _Pred, _Alloc  
>::const_reference
```

Iterator-related typedefs.

Definition at line 114 of file unordered\_set.h.

```
4.713.2.6 template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =  
std::allocator<_Value>> typedef _Hashtable::difference_type std::unordered_set< _Value, _Hash, _Pred, _Alloc  
>::difference_type
```

Iterator-related typedefs.

Definition at line 120 of file unordered\_set.h.

```
4.713.2.7 template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =  
std::allocator<_Value>> typedef _Hashtable::hasher std::unordered_set< _Value, _Hash, _Pred, _Alloc >::hasher
```

Public typedefs.

Definition at line 104 of file unordered\_set.h.

```
4.713.2.8 template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =  
std::allocator<_Value>> typedef _Hashtable::iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::iterator
```

Iterator-related typedefs.

Definition at line 115 of file unordered\_set.h.

```
4.713.2.9 template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =  
std::allocator<_Value>> typedef _Hashtable::key_equal std::unordered_set< _Value, _Hash, _Pred, _Alloc  
>::key_equal
```

Public typedefs.

Definition at line 105 of file unordered\_set.h.

```
4.713.2.10 template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =  
std::allocator<_Value>> typedef _Hashtable::key_type std::unordered_set< _Value, _Hash, _Pred, _Alloc  
>::key_type
```

Public typedefs.

Definition at line 102 of file unordered\_set.h.

4.713.2.11 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::local_iterator`

Iterator-related typedefs.

Definition at line 117 of file unordered\_set.h.

4.713.2.12 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef allocator_type::pointer std::unordered_set< _Value, _Hash, _Pred, _Alloc >::pointer`

Iterator-related typedefs.

Definition at line 111 of file unordered\_set.h.

4.713.2.13 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef allocator_type::reference std::unordered_set< _Value, _Hash, _Pred, _Alloc >::reference`

Iterator-related typedefs.

Definition at line 113 of file unordered\_set.h.

4.713.2.14 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::size_type`

Iterator-related typedefs.

Definition at line 119 of file unordered\_set.h.

4.713.2.15 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::value_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::value_type`

Public typedefs.

Definition at line 103 of file unordered\_set.h.

#### 4.713.3 Constructor & Destructor Documentation

4.713.3.1 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set( size_type __n = 10, const hasher & __hf = hasher(), const key_equal & __eqf = key_equal(), const allocator_type & __a = allocator_type() ) [inline], [explicit]`

Default constructor creates no elements.

Parameters

<code>__n</code>	Initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eqf</code>	A key equality functor.

<code>__a</code>	An allocator object.
------------------	----------------------

Definition at line 132 of file unordered\_set.h.

```
4.713.3.2 template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> template<typename _InputIterator > std::unordered_set< _Value, _Hash, _Pred, _Alloc
>::unordered_set ( _InputIterator __f, _InputIterator __l, size_type __n = 0, const hasher & __hf = hasher(),
const key_equal & __eqf = key_equal(), const allocator_type & __a = allocator_type() ) [inline]
```

Builds an unordered\_set from a range.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__n</code>	Minimal initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eqf</code>	A key equality functor.
<code>__a</code>	An allocator object.

Create an unordered\_set consisting of copies of the elements from [`__first`,`__last`). This is linear in N (where N is `distance(__first, __last)`).

Definition at line 153 of file unordered\_set.h.

```
4.713.3.3 template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set ( const
unordered_set< _Value, _Hash, _Pred, _Alloc > & ) [default]
```

Copy constructor.

```
4.713.3.4 template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
unordered_set< _Value, _Hash, _Pred, _Alloc > && ) [default]
```

Move constructor.

```
4.713.3.5 template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set ( initializer_list<
value_type > __l, size_type __n = 0, const hasher & __hf = hasher(), const key_equal & __eqf =
key_equal(), const allocator_type & __a = allocator_type() ) [inline]
```

Builds an unordered\_set from an initializer\_list.

Parameters

<code>__l</code>	An initializer_list.
<code>__n</code>	Minimal initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eqf</code>	A key equality functor.
<code>__a</code>	An allocator object.

Create an unordered\_set consisting of copies of the elements in the list. This is linear in N (where N is `__l.size()`).

Definition at line 178 of file unordered\_set.h.

#### 4.713.4 Member Function Documentation

4.713.4.1 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::begin ( ) [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the `unordered_set`.

Definition at line 243 of file `unordered_set.h`.

4.713.4.2 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::begin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the `unordered_set`.

Definition at line 247 of file `unordered_set.h`.

4.713.4.3 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::begin ( size_type __n ) [inline]`

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

<code>__n</code>	The bucket index.
------------------	-------------------

Returns

A read-only local iterator.

Definition at line 593 of file `unordered_set.h`.

4.713.4.4 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::begin ( size_type __n ) const [inline]`

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

<code>__n</code>	The bucket index.
------------------	-------------------

Returns

A read-only local iterator.

Definition at line 597 of file `unordered_set.h`.

4.713.4.5 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> size_type std::unordered_set<_Value, _Hash, _Pred, _Alloc>::bucket_count ( ) const [inline], [noexcept]`

Returns the number of buckets of the `unordered_set`.

Definition at line 559 of file `unordered_set.h`.

4.713.4.6 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc >::cbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the unordered\_set.

Definition at line 270 of file unordered\_set.h.

4.713.4.7 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc >::cbegin ( size_type __n ) const [inline]`

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

<code>__n</code>	The bucket index.
------------------	-------------------

Returns

A read-only local iterator.

Definition at line 601 of file unordered\_set.h.

4.713.4.8 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc >::cend ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the unordered\_set.

Definition at line 278 of file unordered\_set.h.

4.713.4.9 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc >::cend ( size_type __n ) const [inline]`

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

<code>__n</code>	The bucket index.
------------------	-------------------

Returns

A read-only local iterator.

Definition at line 621 of file unordered\_set.h.

4.713.4.10 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void std::unordered_set<_Value, _Hash, _Pred, _Alloc >::clear ( ) [inline], [noexcept]`

Erases all elements in an unordered\_set. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 471 of file unordered\_set.h.



4.713.4.11 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =  
std::allocator<_Value>> size_type std::unordered_set<_Value, _Hash, _Pred, _Alloc >::count ( const  
key_type & __x ) const [inline]`

Finds the number of elements.

## Parameters

<code>__x</code>	Element to located.
------------------	---------------------

## Returns

Number of elements with specified key.

This function only makes sense for unordered\_multisets; for unordered\_set the result will either be 0 (not present) or 1 (present).

Definition at line 534 of file unordered\_set.h.

4.713.4.12 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> template<typename... _Args> std::pair<iterator, bool> std::unordered_set<_Value, _Hash, _Pred, _Alloc >::emplace ( _Args &&... __args ) [inline]`

Attempts to build and insert an element into the unordered\_set.

## Parameters

<code>__args</code>	Arguments used to generate an element.
---------------------	--

## Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to build and insert an element into the unordered\_set. An unordered\_set relies on unique keys and thus an element is only inserted if it is not already present in the unordered\_set.

Insertion requires amortized constant time.

Definition at line 300 of file unordered\_set.h.

4.713.4.13 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> template<typename... _Args> iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc >::emplace_hint ( const_iterator __pos, _Args &&... __args ) [inline]`

Attempts to insert an element into the unordered\_set.

## Parameters

<code>__pos</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__args</code>	Arguments used to generate the element to be inserted.

## Returns

An iterator that points to the element with key equivalent to the one generated from `__args` (may or may not be the element itself).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `emplace()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html>

Insertion requires amortized constant time.

Definition at line 326 of file unordered\_set.h.

4.713.4.14 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> bool std::unordered_set<_Value, _Hash, _Pred, _Alloc>::empty ( ) const [inline], [noexcept]`

Returns true if the `unordered_set` is empty.

Definition at line 222 of file `unordered_set.h`.

4.713.4.15 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::end ( ) [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the `unordered_set`.

Definition at line 257 of file `unordered_set.h`.

4.713.4.16 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::end ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the `unordered_set`.

Definition at line 261 of file `unordered_set.h`.

4.713.4.17 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::end ( size_type __n ) [inline]`

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

<code>__n</code>	The bucket index.
------------------	-------------------

Returns

A read-only local iterator.

Definition at line 613 of file `unordered_set.h`.

4.713.4.18 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::end ( size_type __n ) const [inline]`

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

<code>__n</code>	The bucket index.
------------------	-------------------

**Returns**

A read-only local iterator.

Definition at line 617 of file `unordered_set.h`.

```
4.713.4.19 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =  
std::allocator<_Value>> std::pair<iterator, iterator> std::unordered_set<_Value, _Hash, _Pred, _Alloc  
>::equal_range( const key_type &__x ) [inline]
```

Finds a subsequence matching given key.

**Parameters**

<code>__x</code>	Key to be located.
------------------	--------------------

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for multisets.

Definition at line 547 of file `unordered_set.h`.

```
4.713.4.20 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> std::pair<const_iterator, const_iterator> std::unordered_set<_Value, _Hash,
_Pred, _Alloc>::equal_range ( const key_type & __x ) const [inline]
```

Finds a subsequence matching given key.

**Parameters**

<code>__x</code>	Key to be located.
------------------	--------------------

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for multisets.

Definition at line 551 of file `unordered_set.h`.

```
4.713.4.21 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::erase ( const_iterator
__position ) [inline]
```

Erases an element from an `unordered_set`.

**Parameters**

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an `unordered_set`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 421 of file `unordered_set.h`.

```
4.713.4.22 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::erase ( iterator __it )
[inline]
```

Erases an element from an `unordered_set`.

## Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

## Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an `unordered_set`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 426 of file `unordered_set.h`.

```
4.713.4.23 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> size_type std::unordered_set<_Value, _Hash, _Pred, _Alloc>::erase ( const
key_type & __x ) [inline]
```

Erases elements according to the provided key.

## Parameters

<code>__x</code>	Key of element to be erased.
------------------	------------------------------

## Returns

The number of elements erased.

This function erases all the elements located by the given key from an `unordered_set`. For an `unordered_set` the result of this function can only be 0 (not present) or 1 (present). Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 443 of file `unordered_set.h`.

```
4.713.4.24 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::erase ( const_iterator
__first, const_iterator __last ) [inline]
```

Erases a [`__first`,`__last`) range of elements from an `unordered_set`.

## Parameters

<code>__first</code>	Iterator pointing to the start of the range to be erased.
<code>__last</code>	Iterator pointing to the end of the range to be erased.

## Returns

The iterator `__last`.

This function erases a sequence of elements from an `unordered_set`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 461 of file `unordered_set.h`.

4.713.4.25 `template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =  
std::allocator<_Value>> iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc >::find ( const key_type &  
__x ) [inline]`

Tries to locate an element in an `unordered_set`.

## Parameters

<code>__x</code>	Element to be located.
------------------	------------------------

## Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 516 of file `unordered_set.h`.

```
4.713.4.26 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> const_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::find( const
key_type & __x ) const [inline]
```

Tries to locate an element in an `unordered_set`.

## Parameters

<code>__x</code>	Element to be located.
------------------	------------------------

## Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 520 of file `unordered_set.h`.

```
4.713.4.27 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> allocator_type std::unordered_set<_Value, _Hash, _Pred, _Alloc>::get_allocator( )
const [inline], [noexcept]
```

Returns the allocator object with which the `unordered_set` was constructed.

Definition at line 215 of file `unordered_set.h`.

```
4.713.4.28 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> hasher std::unordered_set<_Value, _Hash, _Pred, _Alloc>::hash_function( ) const
[inline]
```

Returns the hash functor object with which the `unordered_set` was constructed.

Definition at line 492 of file `unordered_set.h`.

```
4.713.4.29 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> std::pair<iterator, bool> std::unordered_set<_Value, _Hash, _Pred, _Alloc>::insert
( const value_type & __x ) [inline]
```

Attempts to insert an element into the `unordered_set`.



**Parameters**

<code>__x</code>	Element to be inserted.
------------------	-------------------------

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the `unordered_set`. An `unordered_set` relies on unique keys and thus an element is only inserted if it is not already present in the `unordered_set`.

Insertion requires amortized constant time.

Definition at line 344 of file `unordered_set.h`.

```
4.713.4.30 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> std::pair<iterator, bool> std::unordered_set<_Value, _Hash, _Pred, _Alloc>::insert
( value_type && __x ) [inline]
```

Attempts to insert an element into the `unordered_set`.

**Parameters**

<code>__x</code>	Element to be inserted.
------------------	-------------------------

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the `unordered_set`. An `unordered_set` relies on unique keys and thus an element is only inserted if it is not already present in the `unordered_set`.

Insertion requires amortized constant time.

Definition at line 348 of file `unordered_set.h`.

References `std::move()`.

```
4.713.4.31 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::insert ( const_iterator
__hint, const value_type & __x ) [inline]
```

Attempts to insert an element into the `unordered_set`.

**Parameters**

<code>__hint</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__x</code>	Element to be inserted.

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17>.↵  
html

Insertion requires amortized constant.

Definition at line 373 of file unordered\_set.h.

```
4.713.4.32 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc >::insert ( const_iterator
__hint, value_type && __x ) [inline]
```

Attempts to insert an element into the unordered\_set.

#### Parameters

<code>__hint</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__x</code>	Element to be inserted.

#### Returns

An iterator that points to the element with key of `__x` (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17>.↵  
html

Insertion requires amortized constant.

Definition at line 377 of file unordered\_set.h.

References `std::move()`.

```
4.713.4.33 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> template<typename _InputIterator > void std::unordered_set<_Value, _Hash, _Pred,
_Alloc >::insert ( _InputIterator __first, _InputIterator __last ) [inline]
```

A template function that attempts to insert a range of elements.

#### Parameters

<code>__first</code>	Iterator pointing to the start of the range to be inserted.
<code>__last</code>	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 392 of file unordered\_set.h.

```
4.713.4.34 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> void std::unordered_set<_Value, _Hash, _Pred, _Alloc >::insert ( initializer_list<
value_type > __l ) [inline]
```

Attempts to insert a list of elements into the unordered\_set.

#### Parameters

<code>__l</code>	A <code>std::initializer_list&lt;value_type&gt;</code> of elements to be inserted.
------------------	--

Complexity similar to that of the range constructor.

Definition at line 403 of file `unordered_set.h`.

```
4.713.4.35 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> key_equal std::unordered_set<_Value, _Hash, _Pred, _Alloc>::key_eq ( ) const
[inline]
```

Returns the key comparison object with which the `unordered_set` was constructed.

Definition at line 498 of file `unordered_set.h`.

```
4.713.4.36 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> float std::unordered_set<_Value, _Hash, _Pred, _Alloc>::load_factor ( ) const
[inline], [noexcept]
```

Returns the average number of elements per bucket.

Definition at line 629 of file `unordered_set.h`.

```
4.713.4.37 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> size_type std::unordered_set<_Value, _Hash, _Pred, _Alloc>::max_bucket_count ( )
const [inline], [noexcept]
```

Returns the maximum number of buckets of the `unordered_set`.

Definition at line 564 of file `unordered_set.h`.

```
4.713.4.38 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> float std::unordered_set<_Value, _Hash, _Pred, _Alloc>::max_load_factor ( ) const
[inline], [noexcept]
```

Returns a positive number that the `unordered_set` tries to keep the load factor less than or equal to.

Definition at line 635 of file `unordered_set.h`.

```
4.713.4.39 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> void std::unordered_set<_Value, _Hash, _Pred, _Alloc>::max_load_factor ( float __z )
[inline]
```

Change the `unordered_set` maximum load factor.

Parameters

<code>__z</code>	The new maximum load factor.
------------------	------------------------------

Definition at line 643 of file `unordered_set.h`.

```
4.713.4.40 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> size_type std::unordered_set<_Value, _Hash, _Pred, _Alloc>::max_size ( ) const
[inline], [noexcept]
```

Returns the maximum size of the `unordered_set`.

Definition at line 232 of file `unordered_set.h`.

4.713.4.41 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> unordered_set& std::unordered_set<_Value, _Hash, _Pred, _Alloc>::operator= (const unordered_set<_Value, _Hash, _Pred, _Alloc> & ) [default]`

Copy assignment operator.

4.713.4.42 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> unordered_set& std::unordered_set<_Value, _Hash, _Pred, _Alloc>::operator= (unordered_set<_Value, _Hash, _Pred, _Alloc> && ) [default]`

Move assignment operator.

4.713.4.43 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> unordered_set& std::unordered_set<_Value, _Hash, _Pred, _Alloc>::operator= (initializer_list<value_type> __l ) [inline]`

Unordered\_set list assignment operator.

Parameters

<code>__l</code>	An initializer_list.
------------------	----------------------

This function fills an unordered\_set with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the unordered\_set and that the resulting unordered\_set's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 206 of file unordered\_set.h.

4.713.4.44 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void std::unordered_set<_Value, _Hash, _Pred, _Alloc>::rehash ( size_type __n ) [inline]`

May rehash the unordered\_set.

Parameters

<code>__n</code>	The new number of buckets.
------------------	----------------------------

Rehash will occur only if the new number of buckets respect the unordered\_set maximum load factor.

Definition at line 654 of file unordered\_set.h.

4.713.4.45 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void std::unordered_set<_Value, _Hash, _Pred, _Alloc>::reserve ( size_type __n ) [inline]`

Prepare the unordered\_set for a specified number of elements.

Parameters

<code>__n</code>	Number of elements required.
------------------	------------------------------

Same as `rehash(ceil(n / max_load_factor()))`.

Definition at line 665 of file unordered\_set.h.

4.713.4.46 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> size_type std::unordered_set<_Value, _Hash, _Pred, _Alloc>::size ( ) const [inline], [noexcept]`

Returns the size of the unordered\_set.

Definition at line 227 of file unordered\_set.h.

```
4.713.4.47  template<class _Value , class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::swap ( unordered_set<
            _Value, _Hash, _Pred, _Alloc > &__x )  [inline]
```

Swaps data with another unordered\_set.

Parameters

__x	An unordered_set of the same element and allocator types.
-----	---

This exchanges the elements between two sets in constant time. Note that the global std::swap() function is specialized such that std::swap(s1,s2) will feed to this function.

Definition at line 484 of file unordered\_set.h.

The documentation for this class was generated from the following file:

- [unordered\\_set.h](#)

## 4.714 std::uses\_allocator< \_Tp, \_Alloc > Struct Template Reference

Inherits integral\_constant< bool, \_\_uses\_allocator\_helper< \_Tp, \_Alloc >::value >.

### 4.714.1 Detailed Description

```
template<typename _Tp, typename _Alloc>struct std::uses_allocator< _Tp, _Alloc >
```

Declare uses\_allocator so it can be specialized in <queue> etc.

[allocator.uses.trait]

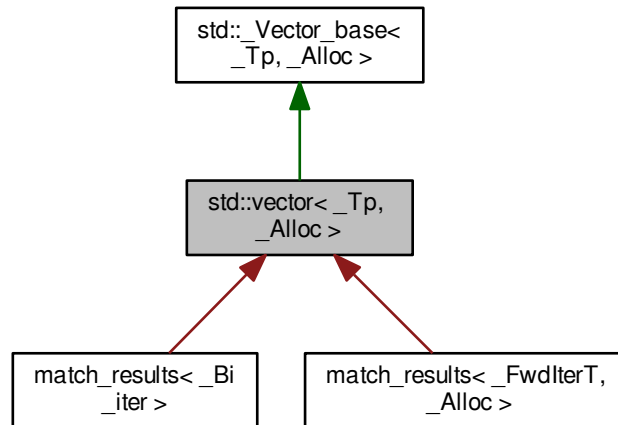
Definition at line 71 of file memoryfwd.h.

The documentation for this struct was generated from the following file:

- [memoryfwd.h](#)

## 4.715 std::vector&lt; \_Tp, \_Alloc &gt; Class Template Reference

Inheritance diagram for std::vector< \_Tp, \_Alloc >:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `__gnu_cxx::__normal_iterator< const_pointer, vector >` **const\_iterator**
- typedef `_Alloc_traits::const_pointer` **const\_pointer**
- typedef `_Alloc_traits::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `__gnu_cxx::__normal_iterator< pointer, vector >` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Alloc_traits::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- `vector()`
- `vector(const allocator_type &__a)`
- `vector(size_type __n, const allocator_type &__a=allocator_type())`
- `vector(size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())`
- `vector(const vector &__x)`
- `vector(vector &&__x) noexcept`
- `vector(const vector &__x, const allocator_type &__a)`
- `vector(vector &&__rv, const allocator_type &__m)`

- [vector](#) (initializer\_list< value\_type > \_\_l, const allocator\_type & \_\_a=allocator\_type())
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>> [vector](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type & \_\_a=allocator\_type())
- [~vector](#) () noexcept
- void [assign](#) (size\_type \_\_n, const value\_type & \_\_val)
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>> void [assign](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void [assign](#) (initializer\_list< value\_type > \_\_l)
- reference [at](#) (size\_type \_\_n)
- const\_reference [at](#) (size\_type \_\_n) const
- reference [back](#) ()
- const\_reference [back](#) () const
- iterator [begin](#) () noexcept
- const\_iterator [begin](#) () const noexcept
- size\_type [capacity](#) () const noexcept
- const\_iterator [cbegin](#) () const noexcept
- const\_iterator [cend](#) () const noexcept
- void [clear](#) () noexcept
- const\_reverse\_iterator [crbegin](#) () const noexcept
- const\_reverse\_iterator [crend](#) () const noexcept
- \_Tp \* [data](#) () noexcept
- const \_Tp \* [data](#) () const noexcept
- template<typename... \_Args> iterator [emplace](#) (iterator \_\_position, \_Args &&... \_\_args)
- template<typename... \_Args> void [emplace\\_back](#) (\_Args &&... \_\_args)
- bool [empty](#) () const noexcept
- iterator [end](#) () noexcept
- const\_iterator [end](#) () const noexcept
- iterator [erase](#) (iterator \_\_position)
- iterator [erase](#) (iterator \_\_first, iterator \_\_last)
- reference [front](#) ()
- const\_reference [front](#) () const
- iterator [insert](#) (iterator \_\_position, const value\_type & \_\_x)
- iterator [insert](#) (iterator \_\_position, value\_type && \_\_x)
- void [insert](#) (iterator \_\_position, initializer\_list< value\_type > \_\_l)
- void [insert](#) (iterator \_\_position, size\_type \_\_n, const value\_type & \_\_x)
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>> void [insert](#) (iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- size\_type [max\\_size](#) () const noexcept
- [vector](#) & [operator=](#) (const [vector](#) & \_\_x)
- [vector](#) & [operator=](#) ([vector](#) && \_\_x) noexcept(\_Alloc\_traits::\_S\_nothrow\_move())
- [vector](#) & [operator=](#) (initializer\_list< value\_type > \_\_l)
- reference [operator\[\]](#) (size\_type \_\_n)
- const\_reference [operator\[\]](#) (size\_type \_\_n) const
- void [pop\\_back](#) ()
- void [push\\_back](#) (const value\_type & \_\_x)
- void [push\\_back](#) (value\_type && \_\_x)
- reverse\_iterator [rbegin](#) () noexcept
- const\_reverse\_iterator [rbegin](#) () const noexcept
- reverse\_iterator [rend](#) () noexcept
- const\_reverse\_iterator [rend](#) () const noexcept
- void [reserve](#) (size\_type \_\_n)

- void [resize](#) (size\_type \_\_new\_size)
- void [resize](#) (size\_type \_\_new\_size, const value\_type &\_\_x)
- void [shrink\\_to\\_fit](#) ()
- size\_type [size](#) () const noexcept
- void [swap](#) (vector &\_\_x) noexcept(\_Alloc\_traits::\_S\_nothrow\_swap())

### Protected Member Functions

- pointer [\\_M\\_allocate](#) (size\_t \_\_n)
- template<typename \_ForwardIterator> pointer [\\_M\\_allocate\\_and\\_copy](#) (size\_type \_\_n, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last)
- template<typename \_InputIterator> void [\\_M\\_assign\\_aux](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- template<typename \_ForwardIterator> void [\\_M\\_assign\\_aux](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- template<typename \_Integer> void [\\_M\\_assign\\_dispatch](#) (\_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- template<typename \_InputIterator> void [\\_M\\_assign\\_dispatch](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- size\_type [\\_M\\_check\\_len](#) (size\_type \_\_n, const char \*\_\_s) const
- void [\\_M\\_deallocate](#) (pointer \_\_p, size\_t \_\_n)
- void [\\_M\\_default\\_append](#) (size\_type \_\_n)
- void [\\_M\\_default\\_initialize](#) (size\_type \_\_n)
- template<typename... \_Args> void [\\_M\\_emplace\\_back\\_aux](#) (\_Args &&...\_\_args)
- void [\\_M\\_erase\\_at\\_end](#) (pointer \_\_pos)
- void [\\_M\\_fill\\_assign](#) (size\_type \_\_n, const value\_type &\_\_val)
- void [\\_M\\_fill\\_initialize](#) (size\_type \_\_n, const value\_type &\_\_value)
- void [\\_M\\_fill\\_insert](#) (iterator \_\_pos, size\_type \_\_n, const value\_type &\_\_x)
- \_Tp\_alloc\_type & [\\_M\\_get\\_Tp\\_allocator](#) () noexcept
- const \_Tp\_alloc\_type & [\\_M\\_get\\_Tp\\_allocator](#) () const noexcept
- template<typename \_Integer> void [\\_M\\_initialize\\_dispatch](#) (\_Integer \_\_n, \_Integer \_\_value, \_\_true\_type)
- template<typename \_InputIterator> void [\\_M\\_initialize\\_dispatch](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- template<typename... \_Args> void [\\_M\\_insert\\_aux](#) (iterator \_\_position, \_Args &&...\_\_args)
- template<typename \_Integer> void [\\_M\\_insert\\_dispatch](#) (iterator \_\_pos, \_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- template<typename \_InputIterator> void [\\_M\\_insert\\_dispatch](#) (iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- void [\\_M\\_range\\_check](#) (size\_type \_\_n) const
- template<typename \_InputIterator> void [\\_M\\_range\\_initialize](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- template<typename \_ForwardIterator> void [\\_M\\_range\\_initialize](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- template<typename \_InputIterator> void [\\_M\\_range\\_insert](#) (iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- template<typename \_ForwardIterator> void [\\_M\\_range\\_insert](#) (iterator \_\_pos, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- bool [\\_M\\_shrink\\_to\\_fit](#) ()
- allocator\_type [get\\_allocator](#) () const noexcept

### Protected Attributes

- [\\_Vector\\_impl](#) [\\_M\\_impl](#)



## 4.715.1 Detailed Description

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>> class std::vector< _Tp, _Alloc >
```

A standard container which offers fixed time access to individual elements in any order.

## Template Parameters

<code>_Tp</code>	Type of element.
<code>_Alloc</code>	Allocator type, defaults to <code>allocator&lt;_Tp&gt;</code> .

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#) with the exception of `push_front` and `pop_front`.

In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from worrying about memory and size allocation. Subscripting ( `[]` ) access is also provided as with C-style arrays.

Definition at line 210 of file `stl_vector.h`.

## 4.715.2 Constructor &amp; Destructor Documentation

```
4.715.2.1 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector< _Tp, _Alloc >::vector ( )
[inline]
```

Default constructor creates no elements.

Definition at line 248 of file `stl_vector.h`.

```
4.715.2.2 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector< _Tp, _Alloc >::vector ( const
allocator_type &__a ) [inline],[explicit]
```

Creates a vector with no elements.

## Parameters

<code>__a</code>	An allocator object.
------------------	----------------------

Definition at line 256 of file `stl_vector.h`.

```
4.715.2.3 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector< _Tp, _Alloc >::vector ( size_type
__n, const allocator_type &__a=allocator_type() ) [inline],[explicit]
```

Creates a vector with default constructed elements.

## Parameters

<code>__n</code>	The number of elements to initially create.
<code>__a</code>	An allocator.

This constructor fills the vector with `__n` default constructed elements.

Definition at line 269 of file `stl_vector.h`.

```
4.715.2.4 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector< _Tp, _Alloc >::vector ( size_type
__n, const value_type &__value, const allocator_type &__a=allocator_type() ) [inline]
```

Creates a vector with copies of an exemplar element.

## Parameters

<code>__n</code>	The number of elements to initially create.
<code>__value</code>	An element to copy.
<code>__a</code>	An allocator.

This constructor fills the vector with `__n` copies of `__value`.

Definition at line 281 of file `stl_vector.h`.

**4.715.2.5** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector< _Tp, _Alloc >::vector ( const vector< _Tp, _Alloc > & __x ) [inline]`

Vector copy constructor.

## Parameters

<code>__x</code>	A vector of identical element and allocator types.
------------------	--

The newly-created vector uses a copy of the allocation object used by `__x`. All the elements of `__x` are copied, but any extra memory in `__x` (for fast expansion) will not be copied.

Definition at line 310 of file `stl_vector.h`.

**4.715.2.6** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector< _Tp, _Alloc >::vector ( vector< _Tp, _Alloc > && __x ) [inline], [noexcept]`

Vector move constructor.

## Parameters

<code>__x</code>	A vector of identical element and allocator types.
------------------	--

The newly-created vector contains the exact contents of `__x`. The contents of `__x` are a valid, but unspecified vector.

Definition at line 327 of file `stl_vector.h`.

**4.715.2.7** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector< _Tp, _Alloc >::vector ( const vector< _Tp, _Alloc > & __x, const allocator_type & __a ) [inline]`

Copy constructor with alternative allocator.

Definition at line 331 of file `stl_vector.h`.

**4.715.2.8** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector< _Tp, _Alloc >::vector ( vector< _Tp, _Alloc > && __rv, const allocator_type & __m ) [inline]`

Move constructor with alternative allocator.

Definition at line 340 of file `stl_vector.h`.

**4.715.2.9** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector< _Tp, _Alloc >::vector ( initializer_list< value_type > __l, const allocator_type & __a = allocator_type() ) [inline]`

Builds a vector from an initializer list.

## Parameters

<code>__l</code>	An <code>initializer_list</code> .
------------------	------------------------------------

<code>__a</code>	An allocator.
------------------	---------------

Create a vector consisting of copies of the elements in the initializer\_list `__l`.

This will call the element type's copy constructor  $N$  times (where  $N$  is `__l.size()`) and do no memory reallocation.

Definition at line 364 of file `stl_vector.h`.

**4.715.2.10** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>> std::vector<_Tp, _Alloc>::vector ( _InputIterator __first, _InputIterator __last, const allocator_type & __a = allocator_type() ) [inline]`

Builds a vector from a range.

#### Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__a</code>	An allocator.

Create a vector consisting of copies of the elements from `[first,last)`.

If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor  $N$  times (where  $N$  is `distance(first,last)`) and do no memory reallocation. But if only input iterators are used, then this will do at most  $2N$  calls to the copy constructor, and  $\log N$  memory reallocations.

Definition at line 392 of file `stl_vector.h`.

**4.715.2.11** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector<_Tp, _Alloc>::~~vector ( ) [inline], [noexcept]`

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 414 of file `stl_vector.h`.

### 4.715.3 Member Function Documentation

**4.715.3.1** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _ForwardIterator> pointer std::vector<_Tp, _Alloc>::M_allocate_and_copy ( size_type __n, _ForwardIterator __first, _ForwardIterator __last ) [inline], [protected]`

Memory expansion handler. Uses the member allocation function to obtain  $n$  bytes of memory, and then copies `[first,last)` into it.

Definition at line 1135 of file `stl_vector.h`.

**4.715.3.2** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector<_Tp, _Alloc>::M_range_check ( size_type __n ) const [inline], [protected]`

Safety check used only from `at()`.

Definition at line 791 of file `stl_vector.h`.

Referenced by `std::vector<_State>::at()`.

**4.715.3.3** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector<_Tp, _Alloc>::assign ( size_type __n, const value_type & __val ) [inline]`

Assigns a given value to a vector.

## Parameters

<code>__n</code>	Number of elements to be assigned.
<code>__val</code>	Value to be assigned.

This function fills a vector with `__n` copies of the given value. Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 479 of file `stl_vector.h`.

Referenced by `std::vector< _State >::assign()`, and `std::vector< _State >::operator=()`.

**4.715.3.4** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>> void std::vector< _Tp, _Alloc >::assign ( _InputIterator __first, _InputIterator __last ) [inline]`

Assigns a range to a vector.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

This function fills a vector with copies of the elements in the range `[__first,__last)`.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 498 of file `stl_vector.h`.

**4.715.3.5** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector< _Tp, _Alloc >::assign ( initializer_list< value_type > __l ) [inline]`

Assigns an initializer list to a vector.

## Parameters

<code>__l</code>	An <code>initializer_list</code> .
------------------	------------------------------------

This function fills a vector with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 524 of file `stl_vector.h`.

**4.715.3.6** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::vector< _Tp, _Alloc >::at ( size_type __n ) [inline]`

Provides access to the data contained in the vector.

## Parameters

<code>__n</code>	The index of the element for which data should be accessed.
------------------	---

## Returns

Read/write reference to data.

**Exceptions**

<i>std::out_of_range</i>	If <code>__n</code> is an invalid index.
--------------------------	--

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws `out_of_range` if the check fails.

Definition at line 810 of file `stl_vector.h`.

**4.715.3.7** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::vector<_Tp, _Alloc>::at (size_type __n) const [inline]`

Provides access to the data contained in the vector.

**Parameters**

<code>__n</code>	The index of the element for which data should be accessed.
------------------	---

**Returns**

Read-only (constant) reference to data.

**Exceptions**

<i>std::out_of_range</i>	If <code>__n</code> is an invalid index.
--------------------------	--

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws `out_of_range` if the check fails.

Definition at line 828 of file `stl_vector.h`.

**4.715.3.8** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::vector<_Tp, _Alloc>::back ( ) [inline]`

Returns a read/write reference to the data at the last element of the vector.

Definition at line 855 of file `stl_vector.h`.

Referenced by `std::piecewise_constant_distribution<_RealType>::max()`, and `std::piecewise_linear_distribution<_RealType>::max()`.

**4.715.3.9** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::vector<_Tp, _Alloc>::back ( ) const [inline]`

Returns a read-only (constant) reference to the data at the last element of the vector.

Definition at line 863 of file `stl_vector.h`.

**4.715.3.10** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::vector<_Tp, _Alloc>::begin ( ) [inline], [noexcept]`

Returns a read/write iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 538 of file `stl_vector.h`.

Referenced by `std::vector<_State>::crend()`, `std::vector<_State>::empty()`, `std::vector<_State>::front()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, `__gnu_parallel::multiway_merge_exact_splitting()`, `std::operator==( )`, `std::vector<_State>::rend()`, and `std::vector<_State>::vector()`.

**4.715.3.11** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::vector<_Tp, _Alloc>::begin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 547 of file `stl_vector.h`.

**4.715.3.12** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::vector<_Tp, _Alloc>::capacity ( ) const [inline], [noexcept]`

Returns the total number of elements that the vector can hold before needing to allocate more memory.

Definition at line 725 of file `stl_vector.h`.

**4.715.3.13** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::vector<_Tp, _Alloc>::cbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 611 of file `stl_vector.h`.

**4.715.3.14** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::vector<_Tp, _Alloc>::cend ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 620 of file `stl_vector.h`.

**4.715.3.15** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector<_Tp, _Alloc>::clear ( ) [inline], [noexcept]`

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1125 of file `stl_vector.h`.

**4.715.3.16** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::vector<_Tp, _Alloc>::crbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 629 of file `stl_vector.h`.

**4.715.3.17** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::vector<_Tp, _Alloc>::crend ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 638 of file `stl_vector.h`.

**4.715.3.18** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> _Tp* std::vector<_Tp, _Alloc>::data ( ) [inline], [noexcept]`

Returns a pointer such that `[data(), data() + size())` is a valid range. For a non-empty vector, `data() == &front()`.

Definition at line 878 of file `stl_vector.h`.

**4.715.3.19** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename... _Args> iterator  
std::vector<_Tp, _Alloc>::emplace ( iterator __position, _Args &&... __args )`

Inserts an object in vector before specified iterator.

#### Parameters

<code>__position</code>	An iterator into the vector.
<code>__args</code>	Arguments.

#### Returns

An iterator that points to the inserted data.

This function will insert an object of type `T` constructed with `T(std::forward<Args>(args)...) before the specified location`. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Referenced by `std::vector<_State>::insert()`.

**4.715.3.20** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> bool std::vector<_Tp, _Alloc>::empty ( )  
const [inline], [noexcept]`

Returns true if the vector is empty. (Thus `begin()` would equal `end()`.)

Definition at line 734 of file `stl_vector.h`.

Referenced by `std::piecewise_constant_distribution<_RealType>::densities()`, `std::piecewise_linear_distribution<_RealType>::densities()`, `std::piecewise_constant_distribution<_RealType>::intervals()`, `std::piecewise_linear_distribution<_RealType>::intervals()`, `std::discrete_distribution<_IntType>::max()`, `std::piecewise_constant_distribution<_RealType>::max()`, `std::piecewise_linear_distribution<_RealType>::max()`, `std::piecewise_constant_distribution<_RealType>::min()`, `std::piecewise_linear_distribution<_RealType>::min()`, and `std::discrete_distribution<_IntType>::probabilities()`.

**4.715.3.21** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::vector<_Tp, _Alloc>::end ( )  
[inline], [noexcept]`

Returns a read/write iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 556 of file `stl_vector.h`.

Referenced by `std::vector<_State>::back()`, `std::vector<_State>::cbegin()`, `std::vector<_State>::empty()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, `__gnu_parallel::multiway_merge_exact_splitting()`, `std::operator==()`, `std::vector<_State>::push_back()`, `std::vector<_State>::rbegin()`, `std::vector<_State>::resize()`, and `std::vector<_State>::vector()`.

**4.715.3.22** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::vector<_Tp, _Alloc>::end ( )  
const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 565 of file `stl_vector.h`.

4.715.3.23 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::vector<_Tp, _Alloc >::erase (`  
`iterator __position )`

Remove element at given position.



**Parameters**

<code>__position</code>	Iterator pointing to element to be erased.
-------------------------	--

**Returns**

An iterator pointing to the next element (or end()).

This function will erase the element at the given position and thus shorten the vector by one.

Note This operation could be expensive and if it is frequently used the user should consider using `std::list`. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**4.715.3.24** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::vector<_Tp, _Alloc>::erase (`  
`iterator __first, iterator __last )`

Remove a range of elements.

**Parameters**

<code>__first</code>	Iterator pointing to the first element to be erased.
<code>__last</code>	Iterator pointing to one past the last element to be erased.

**Returns**

An iterator pointing to the element pointed to by `__last` prior to erasing (or end()).

This function will erase the elements in the range `[__first,__last)` and shorten the vector accordingly.

Note This operation could be expensive and if it is frequently used the user should consider using `std::list`. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**4.715.3.25** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::vector<_Tp, _Alloc>::front ( )`  
`[inline]`

Returns a read/write reference to the data at the first element of the vector.

Definition at line 839 of file `stl_vector.h`.

Referenced by `std::vector<_State>::data()`, `std::piecewise_constant_distribution<_RealType>::min()`, and `std::piecewise_linear_distribution<_RealType>::min()`.

**4.715.3.26** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::vector<_Tp, _Alloc>::front ( ) const` `[inline]`

Returns a read-only (constant) reference to the data at the first element of the vector.

Definition at line 847 of file `stl_vector.h`.

**4.715.3.27** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::vector<_Tp, _Alloc>::insert (`  
`iterator __position, const value_type & __x )`

Inserts given value into vector before specified iterator.

## Parameters

<code>__position</code>	An iterator into the vector.
<code>__x</code>	Data to be inserted.

## Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Referenced by `std::vector< _State >::insert()`, and `std::vector< _State >::resize()`.

**4.715.3.28** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::vector< _Tp, _Alloc >::insert ( iterator __position, value_type && __x ) [inline]`

Inserts given rvalue into vector before specified iterator.

## Parameters

<code>__position</code>	An iterator into the vector.
<code>__x</code>	Data to be inserted.

## Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 988 of file `stl_vector.h`.

**4.715.3.29** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector< _Tp, _Alloc >::insert ( iterator __position, initializer_list< value_type > __l ) [inline]`

Inserts an `initializer_list` into the vector.

## Parameters

<code>__position</code>	An iterator into the vector.
<code>__l</code>	An <code>initializer_list</code> .

This function will insert copies of the data in the `initializer_list l` into the vector before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 1005 of file `stl_vector.h`.

**4.715.3.30** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector< _Tp, _Alloc >::insert ( iterator __position, size_type __n, const value_type & __x ) [inline]`

Inserts a number of copies of given data into the vector.

## Parameters

---

<code>__position</code>	An iterator into the vector.
<code>__n</code>	Number of elements to be inserted.
<code>__x</code>	Data to be inserted.

This function will insert a specified number of copies of the given data before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 1023 of file `stl_vector.h`.

```
4.715.3.31  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator, typename =
std:: _RequireInputIter<_InputIterator>> void std::vector< _Tp, _Alloc >::insert ( iterator __position, _InputIterator
__first, _InputIterator __last ) [inline]
```

Inserts a range into the vector.

Parameters

<code>__position</code>	An iterator into the vector.
<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

This function will insert copies of the data in the range `[__first,__last)` into the vector before the location specified by *pos*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 1044 of file `stl_vector.h`.

```
4.715.3.32  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::vector< _Tp, _Alloc >::max_size (
) const [inline],[noexcept]
```

Returns the `size()` of the largest possible vector.

Definition at line 650 of file `stl_vector.h`.

```
4.715.3.33  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> vector& std::vector< _Tp, _Alloc >::operator= (
const vector< _Tp, _Alloc > & __x )
```

Vector assignment operator.

Parameters

<code>__x</code>	A vector of identical element and allocator types.
------------------	--

All the elements of `__x` are copied, but any extra memory in `__x` (for fast expansion) will not be copied. Unlike the copy constructor, the allocator object is not copied.

```
4.715.3.34  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> vector& std::vector< _Tp, _Alloc >::operator= (
vector< _Tp, _Alloc > && __x ) [inline],[noexcept]
```

Vector move assignment operator.

Parameters

<code>__x</code>	A vector of identical element and allocator types.
------------------	--

The contents of `__x` are moved into this vector (without copying, if the allocators permit it). `__x` is a valid, but unspecified vector.

Definition at line 439 of file `stl_vector.h`.

```
4.715.3.35 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> vector& std::vector<_Tp, _Alloc>::operator= (  
    initializer_list< value_type > __l ) [inline]
```

Vector list assignment operator.

**Parameters**

<code>__l</code>	An initializer_list.
------------------	----------------------

This function fills a vector with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 461 of file `stl_vector.h`.

```
4.715.3.36  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::vector< _Tp, _Alloc >::operator[ ]
            ( size_type __n ) [inline]
```

Subscript access to the data contained in the vector.

**Parameters**

<code>__n</code>	The index of the element for which data should be accessed.
------------------	---

**Returns**

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 770 of file `stl_vector.h`.

```
4.715.3.37  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::vector< _Tp, _Alloc
            >::operator[ ]( size_type __n ) const [inline]
```

Subscript access to the data contained in the vector.

**Parameters**

<code>__n</code>	The index of the element for which data should be accessed.
------------------	---

**Returns**

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 785 of file `stl_vector.h`.

```
4.715.3.38  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector< _Tp, _Alloc >::pop_back ( )
            [inline]
```

Removes last element.

This is a typical stack operation. It shrinks the vector by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before `pop_back()` is called.

Definition at line 937 of file `stl_vector.h`.

```
4.715.3.39  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector< _Tp, _Alloc >::push_back (
            const value_type &__x ) [inline]
```

Add data to the end of the vector.

## Parameters

<code>__x</code>	Data to be added.
------------------	-------------------

This is a typical stack operation. The function creates an element at the end of the vector and assigns the given data to it. Due to the nature of a vector this operation can be done in constant time if the vector has preallocated space available.

Definition at line 901 of file `stl_vector.h`.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

**4.715.3.40** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reverse_iterator std::vector<_Tp, _Alloc>::rbegin ( ) [inline], [noexcept]`

Returns a read/write reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 574 of file `stl_vector.h`.

**4.715.3.41** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::vector<_Tp, _Alloc>::rbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 583 of file `stl_vector.h`.

**4.715.3.42** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reverse_iterator std::vector<_Tp, _Alloc>::rend ( ) [inline], [noexcept]`

Returns a read/write reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 592 of file `stl_vector.h`.

**4.715.3.43** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::vector<_Tp, _Alloc>::rend ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 601 of file `stl_vector.h`.

**4.715.3.44** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector<_Tp, _Alloc>::reserve ( size_type __n )`

Attempt to preallocate enough memory for specified number of elements.

## Parameters

<code>__n</code>	Number of elements required.
------------------	------------------------------

## Exceptions

<code>std::length_error</code>	If <code>n</code> exceeds <code>max_size()</code> .
--------------------------------	---

This function attempts to reserve enough memory for the vector to hold the specified number of elements. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the number of elements that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory

and copying of vector data.

**4.715.3.45** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector<_Tp, _Alloc>::resize ( size_type __new_size ) [inline]`

Resizes the vector to the specified number of elements.

Parameters

<code>__new_size</code>	Number of elements the vector should contain.
-------------------------	---

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise default constructed elements are appended.

Definition at line 664 of file `stl_vector.h`.

Referenced by `__gnu_parallel::__shrink_and_double()`, and `__gnu_parallel::multiway_merge_exact_splitting()`.

**4.715.3.46** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector<_Tp, _Alloc>::resize ( size_type __new_size, const value_type & __x ) [inline]`

Resizes the vector to the specified number of elements.

Parameters

<code>__new_size</code>	Number of elements the vector should contain.
<code>__x</code>	Data with which new elements should be populated.

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise the vector is extended and new elements are populated with given data.

Definition at line 684 of file `stl_vector.h`.

**4.715.3.47** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector<_Tp, _Alloc>::shrink_to_fit ( ) [inline]`

A non-binding request to reduce `capacity()` to `size()`.

Definition at line 716 of file `stl_vector.h`.

**4.715.3.48** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::vector<_Tp, _Alloc>::size ( ) const [inline],[noexcept]`

Returns the number of elements in the vector.

Definition at line 645 of file `stl_vector.h`.

Referenced by `__gnu_parallel::__shrink()`, `__gnu_parallel::__shrink_and_double()`, `std::vector<_State>::_M_range_check()`, `__gnu_parallel::list_partition()`, `std::discrete_distribution<_IntType>::max()`, `std::operator==()`, and `std::vector<_State>::resize()`.

**4.715.3.49** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector<_Tp, _Alloc>::swap ( vector<_Tp, _Alloc> & __x ) [inline],[noexcept]`

Swaps data with another vector.

Parameters

<code>__x</code>	A vector of the same element and allocator types.
------------------	---

This exchanges the elements between two vectors in constant time. (Three pointers, so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(v1,v2)` will feed to this function.

Definition at line 1108 of file `stl_vector.h`.

Referenced by std::swap().

The documentation for this class was generated from the following file:

- [stl\\_vector.h](#)

## 4.716 std::vector< bool, \_Alloc > Class Template Reference

Inherits std::\_Bvector\_base< \_Alloc >.

### Public Types

- typedef \_Alloc **allocator\_type**
- typedef \_Bit\_const\_iterator **const\_iterator**
- typedef const bool \* **const\_pointer**
- typedef bool **const\_reference**
- typedef [std::reverse\\_iterator](#)< const\_iterator > **const\_reverse\_iterator**
- typedef ptrdiff\_t **difference\_type**
- typedef \_Bit\_iterator **iterator**
- typedef \_Bit\_reference \* **pointer**
- typedef \_Bit\_reference **reference**
- typedef [std::reverse\\_iterator](#)< iterator > **reverse\_iterator**
- typedef size\_t **size\_type**
- typedef bool **value\_type**

### Public Member Functions

- **vector** (const allocator\_type &\_\_a)
- **vector** (size\_type \_\_n, const allocator\_type &\_\_a=allocator\_type())
- **vector** (size\_type \_\_n, const bool &\_\_value, const allocator\_type &\_\_a=allocator\_type())
- **vector** (const [vector](#) &\_\_x)
- **vector** ([vector](#) &&\_\_x) noexcept
- **vector** (initializer\_list< bool > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>> **vector** (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a=allocator\_type())
- void **assign** (size\_type \_\_n, const bool &\_\_x)
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>> void **assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void **assign** (initializer\_list< bool > \_\_l)
- reference **at** (size\_type \_\_n)
- const\_reference **at** (size\_type \_\_n) const
- reference **back** ()
- const\_reference **back** () const
- iterator **begin** () noexcept
- const\_iterator **begin** () const noexcept
- size\_type **capacity** () const noexcept
- const\_iterator **cbegin** () const noexcept
- const\_iterator **cend** () const noexcept
- void **clear** () noexcept
- [const\\_reverse\\_iterator](#) **crbegin** () const noexcept



- [const\\_reverse\\_iterator](#) **crend** () const noexcept
- void **data** () noexcept
- bool **empty** () const noexcept
- iterator **end** () noexcept
- const\_iterator **end** () const noexcept
- iterator **erase** (iterator \_\_position)
- iterator **erase** (iterator \_\_first, iterator \_\_last)
- void **flip** () noexcept
- reference **front** ()
- const\_reference **front** () const
- allocator\_type **get\_allocator** () const
- iterator **insert** (iterator \_\_position, const bool &\_\_x=bool())
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>> void **insert** (iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- void **insert** (iterator \_\_position, size\_type \_\_n, const bool &\_\_x)
- void **insert** (iterator \_\_p, initializer\_list< bool > \_\_l)
- size\_type **max\_size** () const noexcept
- [vector](#) & **operator=** (const [vector](#) &\_\_x)
- [vector](#) & **operator=** ([vector](#) &&\_\_x)
- [vector](#) & **operator=** (initializer\_list< bool > \_\_l)
- reference **operator[]** (size\_type \_\_n)
- const\_reference **operator[]** (size\_type \_\_n) const
- void **pop\_back** ()
- void **push\_back** (bool \_\_x)
- [reverse\\_iterator](#) **rbegin** () noexcept
- [const\\_reverse\\_iterator](#) **rbegin** () const noexcept
- [reverse\\_iterator](#) **rend** () noexcept
- [const\\_reverse\\_iterator](#) **rend** () const noexcept
- void **reserve** (size\_type \_\_n)
- void **resize** (size\_type \_\_new\_size, bool \_\_x=bool())
- void **shrink\_to\_fit** ()
- size\_type **size** () const noexcept
- void **swap** ([vector](#) &\_\_x)

#### Static Public Member Functions

- static void **swap** (reference \_\_x, reference \_\_y) noexcept

#### Protected Types

- typedef \_Alloc::template rebind< \_Bit\_type >::other **\_Bit\_alloc\_type**

#### Protected Member Functions

- \_Bit\_type \* **\_M\_allocate** (size\_t \_\_n)
- template<typename \_InputIterator > void **\_M\_assign\_aux** (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- template<typename \_ForwardIterator > void **\_M\_assign\_aux** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- template<typename \_Integer > void **\_M\_assign\_dispatch** (\_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)

- `template<class _InputIterator > void _M_assign_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `size_type _M_check_len (size_type __n, const char *__s) const`
- `iterator _M_copy_aligned (const_iterator __first, const_iterator __last, iterator __result)`
- `void _M_deallocate ()`
- `void _M_erase_at_end (iterator __pos)`
- `void _M_fill_assign (size_t __n, bool __x)`
- `void _M_fill_insert (iterator __position, size_type __n, bool __x)`
- `_Bit_alloc_type & _M_get_Bit_allocator () noexcept`
- `const _Bit_alloc_type & _M_get_Bit_allocator () const noexcept`
- `void _M_initialize (size_type __n)`
- `template<typename _Integer > void _M_initialize_dispatch (_Integer __n, _Integer __x, __true_type)`
- `template<typename _InputIterator > void _M_initialize_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `template<typename _InputIterator > void _M_initialize_range (_InputIterator __first, _InputIterator __last, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator > void _M_initialize_range (_ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `void _M_insert_aux (iterator __position, bool __x)`
- `template<typename _Integer > void _M_insert_dispatch (iterator __pos, _Integer __n, _Integer __x, __true_type)`
- `template<typename _InputIterator > void _M_insert_dispatch (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)`
- `template<typename _InputIterator > void _M_insert_range (iterator __pos, _InputIterator __first, _InputIterator __last, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator > void _M_insert_range (iterator __position, _ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `void _M_range_check (size_type __n) const`
- `void _M_reallocate (size_type __n)`
- `bool _M_shrink_to_fit ()`

#### Static Protected Member Functions

- `static size_t _S_nword (size_t __n)`

#### Protected Attributes

- `_Bvector_impl _M_impl`

#### Friends

- `template<typename > struct hash`

#### 4.716.1 Detailed Description

`template<typename _Alloc> class std::vector< bool, _Alloc >`

A specialization of `vector` for booleans which offers fixed time access to individual elements in any order.

## Template Parameters

<code>_Alloc</code>	Allocator type.
---------------------	-----------------

Note that `vector<bool>` does not actually meet the requirements for being a container. This is because the reference and pointer types are not really references and pointers to `bool`. See DR96 for details.

## See also

`vector` for function documentation.

In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from worrying about memory and size allocation. Subscripting ( `[]` ) access is also provided as with C-style arrays.

Definition at line 518 of file `stl_bvector.h`.

The documentation for this class was generated from the following file:

- [stl\\_bvector.h](#)

4.717 `std::weak_ptr<_Tp>` Class Template Reference

Inherits `std::__weak_ptr<_Tp, _Lp>`.

## Public Types

- `typedef _Tp element_type`

## Public Member Functions

- `template<typename _Tp1, typename = typename std::enable_if<std::is_convertible<_Tp1*, _Tp*>::value>::type> weak_ptr (const weak_ptr<_Tp1> &__r) noexcept`
- `template<typename _Tp1, typename = typename std::enable_if<std::is_convertible<_Tp1*, _Tp*>::value>::type> weak_ptr (const shared_ptr<_Tp1> &__r) noexcept`
- `bool expired () const noexcept`
- `shared_ptr<_Tp> lock () const noexcept`
- `template<typename _Tp1> weak_ptr & operator= (const weak_ptr<_Tp1> &__r) noexcept`
- `template<typename _Tp1> weak_ptr & operator= (const shared_ptr<_Tp1> &__r) noexcept`
- `template<typename _Tp1> bool owner_before (const __shared_ptr<_Tp1, _Lp> &__rhs) const`
- `template<typename _Tp1> bool owner_before (const __weak_ptr<_Tp1, _Lp> &__rhs) const`
- `void reset () noexcept`
- `void swap (__weak_ptr &__s) noexcept`
- `long use_count () const noexcept`

## 4.717.1 Detailed Description

`template<typename _Tp> class std::weak_ptr<_Tp>`

A smart pointer with weak semantics.

With forwarding constructors and assignment operators.

Definition at line 461 of file `shared_ptr.h`.

The documentation for this class was generated from the following file:

- [shared\\_ptr.h](#)

## 4.718 std::weibull\_distribution<\_RealType> Class Template Reference

### Classes

- struct [param\\_type](#)

### Public Types

- typedef \_RealType [result\\_type](#)

### Public Member Functions

- **weibull\_distribution** (\_RealType \_\_a=\_RealType(1), \_RealType \_\_b=\_RealType(1))
- **weibull\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator> void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator> void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator> void **\_\_generate** ([result\\_type](#) \* \_\_f, [result\\_type](#) \* \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- \_RealType **a** () const
- \_RealType **b** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator> [result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator> [result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

### Friends

- bool **operator==** (const [weibull\\_distribution](#) &\_\_d1, const [weibull\\_distribution](#) &\_\_d2)

### 4.718.1 Detailed Description

template<typename \_RealType = double> class std::weibull\_distribution<\_RealType>

A weibull\_distribution random number distribution.

The formula for the normal probability density function is:

$$p(x|\alpha, \beta) = \frac{\alpha}{\beta} \left(\frac{x}{\beta}\right)^{\alpha-1} \exp\left(-\left(\frac{x}{\beta}\right)^{\alpha}\right)$$

Definition at line 4848 of file random.h.

#### 4.718.2 Member Typedef Documentation

##### 4.718.2.1 `template<typename _RealType = double> typedef _RealType std::weibull_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 4851 of file random.h.

#### 4.718.3 Member Function Documentation

##### 4.718.3.1 `template<typename _RealType = double> _RealType std::weibull_distribution< _RealType >::a ( ) const` [inline]

Return the  $a$  parameter of the distribution.

Definition at line 4906 of file random.h.

##### 4.718.3.2 `template<typename _RealType = double> _RealType std::weibull_distribution< _RealType >::b ( ) const` [inline]

Return the  $b$  parameter of the distribution.

Definition at line 4913 of file random.h.

##### 4.718.3.3 `template<typename _RealType = double> result_type std::weibull_distribution< _RealType >::max ( ) const` [inline]

Returns the least upper bound value of the distribution.

Definition at line 4942 of file random.h.

References `std::max()`.

##### 4.718.3.4 `template<typename _RealType = double> result_type std::weibull_distribution< _RealType >::min ( ) const` [inline]

Returns the greatest lower bound value of the distribution.

Definition at line 4935 of file random.h.

##### 4.718.3.5 `template<typename _RealType = double> template<typename UniformRandomNumberGenerator> result_type` `std::weibull_distribution< _RealType >::operator() ( UniformRandomNumberGenerator & __urng )` [inline]

Generating functions.

Definition at line 4950 of file random.h.

##### 4.718.3.6 `template<typename _RealType = double> param_type std::weibull_distribution< _RealType >::param ( ) const` [inline]

Returns the parameter set of the distribution.

Definition at line 4920 of file random.h.

##### 4.718.3.7 `template<typename _RealType = double> void std::weibull_distribution< _RealType >::param ( const` `param_type & __param )` [inline]

Sets the parameter set of the distribution.

## Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 4928 of file `random.h`.

4.718.3.8 `template<typename _RealType = double> void std::weibull_distribution<_RealType>::reset ( ) [inline]`

Resets the distribution state.

Definition at line 4899 of file `random.h`.

## 4.718.4 Friends And Related Function Documentation

4.718.4.1 `template<typename _RealType = double> bool operator== ( const weibull_distribution<_RealType> &__d1, const weibull_distribution<_RealType> &__d2 ) [friend]`

Return true if two Weibull distributions have the same parameters.

Definition at line 4985 of file `random.h`.

The documentation for this class was generated from the following file:

- [random.h](#)

4.719 `std::weibull_distribution<_RealType>::param_type` Struct Reference

## Public Types

- typedef `weibull_distribution<_RealType>` **distribution\_type**

## Public Member Functions

- **param\_type** (`_RealType __a=_RealType(1), _RealType __b=_RealType(1)`)
- `_RealType a () const`
- `_RealType b () const`

## Friends

- bool **operator==** (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

## 4.719.1 Detailed Description

`template<typename _RealType = double> struct std::weibull_distribution<_RealType>::param_type`

Parameter type.

Definition at line 4857 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5 File Documentation

### 5.1 algo.h File Reference

#### Classes

- struct [std::\\_\\_parallel::\\_\\_CRandNumber<\\_MustBeInt>](#)

#### Namespaces

- [std](#)
- [std::\\_\\_parallel](#)

#### Functions

- template<typename \_RAIter > \_RAIter **std::\_\_parallel::\_\_adjacent\_find\_switch** (\_RAIter \_\_begin, \_RAIter \_\_end, random\_access\_iterator\_tag)
- template<typename \_FIterator, typename \_IteratorTag > \_FIterator **std::\_\_parallel::\_\_adjacent\_find\_switch** (\_FIterator \_\_begin, \_FIterator \_\_end, \_IteratorTag)
- template<typename \_FIterator, typename \_BinaryPredicate, typename \_IteratorTag > \_FIterator **std::\_\_parallel::\_\_adjacent\_↵  
\_\_find\_switch** (\_FIterator \_\_begin, \_FIterator \_\_end, \_BinaryPredicate \_\_pred, \_IteratorTag)
- template<typename \_RAIter, typename \_BinaryPredicate > \_RAIter **std::\_\_parallel::\_\_adjacent\_find\_switch** (\_RAIter ↵  
\_\_begin, \_RAIter \_\_end, \_BinaryPredicate \_\_pred, random\_access\_iterator\_tag)
- template<typename \_RAIter, typename \_Predicate > iterator\_traits< \_RAIter >::difference\_type **std::\_\_parallel::\_\_↵  
count\_if\_switch** (\_RAIter \_\_begin, \_RAIter \_\_end, \_Predicate \_\_pred, random\_access\_iterator\_tag, [gnu\\_↵  
\\_\\_parallel::\\_\\_Parallelism](#) \_\_parallelism\_tag=[gnu\\_parallel::parallel\\_unbalanced](#))
- template<typename \_IIter, typename \_Predicate, typename \_IteratorTag > iterator\_traits< \_IIter >::difference\_type **std::\_\_↵  
parallel::\_\_count\_if\_switch** (\_IIter \_\_begin, \_IIter \_\_end, \_Predicate \_\_pred, \_IteratorTag)
- template<typename \_RAIter, typename \_Tp > iterator\_traits< \_RAIter >::difference\_type **std::\_\_parallel::\_\_count\_↵  
switch** (\_RAIter \_\_begin, \_RAIter \_\_end, const \_Tp & \_\_value, random\_access\_iterator\_tag, [gnu\\_parallel::\\_\\_↵  
Parallelism](#) \_\_parallelism\_tag=[gnu\\_parallel::parallel\\_unbalanced](#))
- template<typename \_IIter, typename \_Tp, typename \_IteratorTag > iterator\_traits< \_IIter >::difference\_type **std::\_\_↵  
parallel::\_\_count\_switch** (\_IIter \_\_begin, \_IIter \_\_end, const \_Tp & \_\_value, \_IteratorTag)
- template<typename \_IIter, typename \_FIterator, typename \_IteratorTag1, typename \_IteratorTag2 > \_IIter **std::\_\_parallel::\_\_↵  
\_\_find\_first\_of\_switch** (\_IIter \_\_begin1, \_IIter \_\_end1, \_FIterator \_\_begin2, \_FIterator \_\_end2, \_IteratorTag1, ↵  
\_IteratorTag2)
- template<typename \_RAIter, typename \_FIterator, typename \_BinaryPredicate, typename \_IteratorTag > \_RAIter **std::\_\_parallel\_↵  
::\_\_find\_first\_of\_switch** (\_RAIter \_\_begin1, \_RAIter \_\_end1, \_FIterator \_\_begin2, \_FIterator \_\_end2, \_Binary\_↵  
Predicate \_\_comp, random\_access\_iterator\_tag, \_IteratorTag)
- template<typename \_IIter, typename \_FIterator, typename \_BinaryPredicate, typename \_IteratorTag1, typename \_IteratorTag2 > \_IIter **std::\_\_parallel::\_\_find\_first\_of\_switch** (\_IIter \_\_begin1, \_IIter \_\_end1, \_FIterator \_\_begin2, \_FIterator \_\_end2, ↵  
\_BinaryPredicate \_\_comp, \_IteratorTag1, \_IteratorTag2)
- template<typename \_IIter, typename \_Predicate, typename \_IteratorTag > \_IIter **std::\_\_parallel::\_\_find\_if\_switch** (\_IIter ↵  
\_\_begin, \_IIter \_\_end, \_Predicate \_\_pred, \_IteratorTag)
- template<typename \_RAIter, typename \_Predicate > \_RAIter **std::\_\_parallel::\_\_find\_if\_switch** (\_RAIter \_\_begin, \_R\_↵  
Alter \_\_end, \_Predicate \_\_pred, random\_access\_iterator\_tag)
- template<typename \_IIter, typename \_Tp, typename \_IteratorTag > \_IIter **std::\_\_parallel::\_\_find\_switch** (\_IIter \_\_begin, ↵  
\_IIter \_\_end, const \_Tp & \_\_val, \_IteratorTag)
- template<typename \_RAIter, typename \_Tp > \_RAIter **std::\_\_parallel::\_\_find\_switch** (\_RAIter \_\_begin, \_RAIter \_\_end, ↵  
const \_Tp & \_\_val, random\_access\_iterator\_tag)

- `template<typename _Iter, typename _Function, typename _IteratorTag > _Function std::parallel::for_each_switch (↵  
_Iter __begin, _Iter __end, _Function __f, _IteratorTag)`
- `template<typename _RAIter, typename _Function > _Function std::parallel::for_each_switch (_RAIter __begin,  
_RAIter __end, _Function __f, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag=↵  
gnu\_parallel::parallel\_balanced)`
- `template<typename _OutputIterator, typename _Size, typename _Generator, typename _IteratorTag > _OutputIterator std::↵  
parallel::generate_n_switch (_OutputIterator __begin, _Size __n, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter, typename _Size, typename _Generator > _RAIter std::parallel::generate_n_switch (↵  
_RAIter __begin, _Size __n, _Generator __gen, random_access_iterator_tag, gnu\_parallel::Parallelism __↵  
parallelism_tag= gnu\_parallel::parallel\_balanced)`
- `template<typename _Filterator, typename _Generator, typename _IteratorTag > void std::parallel::generate_switch (↵  
_Filterator __begin, _Filterator __end, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter, typename _Generator > void std::parallel::generate_switch (_RAIter __begin, _RA↵  
Iter __end, _Generator __gen, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag=↵  
gnu\_parallel::parallel\_balanced)`
- `template<typename _Filterator, typename _Compare, typename _IteratorTag > _Filterator std::parallel::max_element↵  
switch (_Filterator __begin, _Filterator __end, _Compare __comp, _IteratorTag)`
- `template<typename _RAIter, typename _Compare > _RAIter std::parallel::max_element_switch (_RAIter __begin,  
_RAIter __end, _Compare __comp, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism↵  
tag= gnu\_parallel::parallel\_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare, typename _IteratorTag1, typename ↵  
_IteratorTag2, typename _IteratorTag3 > _OutputIterator std::parallel::merge_switch (_Iter1 __begin1, _Iter1  
__end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp, _IteratorTag1, _Iterator↵  
Tag2, _IteratorTag3)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare > _OutputIterator std::↵  
parallel::merge_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator  
__result, _Compare __comp, random_access_iterator_tag, random_access_iterator_tag, random_access↵  
iterator_tag)`
- `template<typename _Filterator, typename _Compare, typename _IteratorTag > _Filterator std::parallel::min_element↵  
switch (_Filterator __begin, _Filterator __end, _Compare __comp, _IteratorTag)`
- `template<typename _RAIter, typename _Compare > _RAIter std::parallel::min_element_switch (_RAIter __begin,  
_RAIter __end, _Compare __comp, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism↵  
tag= gnu\_parallel::parallel\_balanced)`
- `template<typename _Filterator, typename _Predicate, typename _IteratorTag > _Filterator std::parallel::partition_switch  
( _Filterator __begin, _Filterator __end, _Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate > _RAIter std::parallel::partition_switch (_RAIter __begin, ↵  
_RAIter __end, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _Filterator, typename _Predicate, typename _Tp, typename _IteratorTag > void std::parallel::replace↵  
_if_switch (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp &__new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate, typename _Tp > void std::parallel::replace_if_switch (_RAIter  
__begin, _RAIter __end, _Predicate __pred, const _Tp &__new_value, random_access_iterator_tag, gnu↵  
\_parallel::Parallelism __parallelism_tag= gnu\_parallel::parallel\_balanced)`
- `template<typename _Filterator, typename _Tp, typename _IteratorTag > void std::parallel::replace_switch (_Filterator  
__begin, _Filterator __end, const _Tp &__old_value, const _Tp &__new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Tp > void std::parallel::replace_switch (_RAIter __begin, _RAIter __end,  
const _Tp &__old_value, const _Tp &__new_value, random_access_iterator_tag, gnu\_parallel::Parallelism ↵  
__parallelism_tag= gnu\_parallel::parallel\_balanced)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BinaryPredicate > _RAIter std::parallel::↵  
search_n_switch (_RAIter __begin, _RAIter __end, _Integer __count, const _Tp &__val, _BinaryPredicate __↵  
binary_pred, random_access_iterator_tag)`
- `template<typename _Filterator, typename _Integer, typename _Tp, typename _BinaryPredicate, typename _IteratorTag > _Filterator  
std::parallel::search_n_switch (_Filterator __begin, _Filterator __end, _Integer __count, const _Tp &__val,  
_BinaryPredicate __binary_pred, _IteratorTag)`



- `template<typename _RAIter1, typename _RAIter2 > _RAIter1 std::parallel::search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _IteratorTag1, typename _IteratorTag2 > _FIterator1 std::parallel::search_switch (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BinaryPredicate > _RAIter1 std::parallel::search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _BinaryPredicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2 > _FIterator1 std::parallel::search_switch (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 > _OutputIterator std::parallel::set_difference_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate > _Output_RAlter std::parallel::set_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 > _OutputIterator std::parallel::set_intersection_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate > _Output_RAlter std::parallel::set_intersection_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 > _OutputIterator std::parallel::set_symmetric_difference_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate > _Output_RAlter std::parallel::set_symmetric_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 > _OutputIterator std::parallel::set_union_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate > _Output_RAlter std::parallel::set_union_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation > _RAIter2 std::parallel::transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation, typename _IteratorTag1, typename _IteratorTag2 > _RAIter2 std::parallel::transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, _IteratorTag1, _IteratorTag2)`

- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BinaryOperation > _RAIter3 std::parallel::transform2_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter3 __result, _BinaryOperation __binary_op, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation, typename _Tag1, typename _Tag2, typename _Tag3 > _OutputIterator std::parallel::transform2_switch ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, _Tag1, _Tag2, _Tag3)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 > _OutputIterator std::parallel::unique_copy_switch ( _Iter __begin, _Iter __last, _OutputIterator __out, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename RandomAccessOutputIterator, typename _Predicate > RandomAccessOutputIterator std::parallel::unique_copy_switch ( _RAIter __begin, _RAIter __last, RandomAccessOutputIterator __out, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filterator > _Filterator std::parallel::adjacent_find ( _Filterator __begin, _Filterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator, typename _BinaryPredicate > _Filterator std::parallel::adjacent_find ( _Filterator __begin, _Filterator __end, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator > _Filterator std::parallel::adjacent_find ( _Filterator __begin, _Filterator __end)`
- `template<typename _Filterator, typename _BinaryPredicate > _Filterator std::parallel::adjacent_find ( _Filterator __begin, _Filterator __end, _BinaryPredicate __pred)`
- `template<typename _Iter, typename _Tp > iterator_traits< _Iter >::difference_type std::parallel::count ( _Iter __begin, _Iter __end, const _Tp & __value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp > iterator_traits< _Iter >::difference_type std::parallel::count ( _Iter __begin, _Iter __end, const _Tp & __value, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp > iterator_traits< _Iter >::difference_type std::parallel::count ( _Iter __begin, _Iter __end, const _Tp & __value)`
- `template<typename _Iter, typename _Predicate > iterator_traits< _Iter >::difference_type std::parallel::count_if ( _Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate > iterator_traits< _Iter >::difference_type std::parallel::count_if ( _Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate > iterator_traits< _Iter >::difference_type std::parallel::count_if ( _Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Tp > _Iter std::parallel::find ( _Iter __begin, _Iter __end, const _Tp & __val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp > _Iter std::parallel::find ( _Iter __begin, _Iter __end, const _Tp & __val)`
- `template<typename _Iter, typename _Filterator > _Iter std::parallel::find_first_of ( _Iter __begin1, _Iter __end1, _Filterator __begin2, _Filterator __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Filterator, typename _BinaryPredicate > _Iter std::parallel::find_first_of ( _Iter __begin1, _Iter __end1, _Filterator __begin2, _Filterator __end2, _BinaryPredicate __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Filterator, typename _BinaryPredicate > _Iter std::parallel::find_first_of ( _Iter __begin1, _Iter __end1, _Filterator __begin2, _Filterator __end2, _BinaryPredicate __comp)`
- `template<typename _Iter, typename _Filterator > _Iter std::parallel::find_first_of ( _Iter __begin1, _Iter __end1, _Filterator __begin2, _Filterator __end2)`
- `template<typename _Iter, typename _Predicate > _Iter std::parallel::find_if ( _Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate > _Iter std::parallel::find_if ( _Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Function > _Function std::parallel::for_each ( _Iter __begin, _Iter __end, _Function __f, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iterator, typename _Function > _Function std::parallel::for_each ( _Iterator __begin, _Iterator __end, _Function __f, \_\_gnu\_parallel::Parallelism __parallelism_tag)`

- `template<typename _Iterator, typename _Function> _Function std::parallel::for_each (_Iterator __begin, _Iterator __end, _Function __f)`
- `template<typename _Filterator, typename _Generator> void std::parallel::generate (_Filterator __begin, _Filterator __end, _Generator __gen, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator, typename _Generator> void std::parallel::generate (_Filterator __begin, _Filterator __end, _Generator __gen, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filterator, typename _Generator> void std::parallel::generate (_Filterator __begin, _Filterator __end, _Generator __gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator> _OutputIterator std::parallel::generate_n (_OutputIterator __begin, _Size __n, _Generator __gen, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator> _OutputIterator std::parallel::generate_n (_OutputIterator __begin, _Size __n, _Generator __gen, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator> _OutputIterator std::parallel::generate_n (_OutputIterator __begin, _Size __n, _Generator __gen)`
- `template<typename _Filterator> _Filterator std::parallel::max_element (_Filterator __begin, _Filterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator, typename _Compare> _Filterator std::parallel::max_element (_Filterator __begin, \_\_gnu\_parallel::sequential\_tag _Filterator __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator> _Filterator std::parallel::max_element (_Filterator __begin, _Filterator __end, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filterator> _Filterator std::parallel::max_element (_Filterator __begin, _Filterator __end)`
- `template<typename _Filterator, typename _Compare> _Filterator std::parallel::max_element (_Filterator __begin, \_\_gnu\_parallel::Parallelism _Filterator __end, _Compare __comp, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filterator, typename _Compare> _Filterator std::parallel::max_element (_Filterator __begin, \_\_gnu\_parallel::Parallelism _Filterator __end, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator> _OutputIterator std::parallel::merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare> _OutputIterator std::parallel::merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare> _OutputIterator std::parallel::merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator> _OutputIterator std::parallel::merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result)`
- `template<typename _Filterator> _Filterator std::parallel::min_element (_Filterator __begin, _Filterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator, typename _Compare> _Filterator std::parallel::min_element (_Filterator __begin, \_\_gnu\_parallel::sequential\_tag _Filterator __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator> _Filterator std::parallel::min_element (_Filterator __begin, _Filterator __end, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filterator> _Filterator std::parallel::min_element (_Filterator __begin, _Filterator __end)`
- `template<typename _Filterator, typename _Compare> _Filterator std::parallel::min_element (_Filterator __begin, \_\_gnu\_parallel::Parallelism _Filterator __end, _Compare __comp, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filterator, typename _Compare> _Filterator std::parallel::min_element (_Filterator __begin, \_\_gnu\_parallel::Parallelism _Filterator __end, _Compare __comp)`
- `template<typename _RAlter> void std::parallel::nth_element (_RAlter __begin, _RAlter __nth, _RAlter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter, typename _Compare> void std::parallel::nth_element (_RAlter __begin, _RAlter __nth, _RAlter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter, typename _Compare> void std::parallel::nth_element (_RAlter __begin, _RAlter __nth, _RAlter __end, _Compare __comp)`

- `template<typename _RAIter> void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end)`
- `template<typename _RAIter, typename _Compare> void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter> void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare> void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter> void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end)`
- `template<typename _FIterator, typename _Predicate> _FIterator std::__parallel::partition (_FIterator __begin, _FIterator __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Predicate> _FIterator std::__parallel::partition (_FIterator __begin, _FIterator __end, _Predicate __pred)`
- `template<typename _RAIter> void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _RandomNumberGenerator> void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rand, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter> void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter, typename _RandomNumberGenerator> void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &&__rand)`
- `template<typename _FIterator, typename _Tp> void std::__parallel::replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Tp> void std::__parallel::replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, \_\_gnu\_parallel::Parallelism\_parallelism\_tag)`
- `template<typename _FIterator, typename _Tp> void std::__parallel::replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _FIterator, typename _Predicate, typename _Tp> void std::__parallel::replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Predicate, typename _Tp> void std::__parallel::replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, \_\_gnu\_parallel::Parallelism\_parallelism\_tag)`
- `template<typename _FIterator, typename _Predicate, typename _Tp> void std::__parallel::replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _FIterator1, typename _FIterator2> _FIterator1 std::__parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator1, typename _FIterator2> _FIterator1 std::__parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate> _FIterator1 std::__parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate> _FIterator1 std::__parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred)`
- `template<typename _FIterator, typename _Integer, typename _Tp> _FIterator std::__parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate> _FIterator std::__parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp> _FIterator std::__parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate> _FIterator std::__parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator> _OutputIterator std::__parallel::set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate> _OutputIterator std::__gnu_parallel::set_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator> _OutputIterator std::__parallel::set_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate> _OutputIterator std::__gnu_parallel::set_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator> _OutputIterator std::__parallel::set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate> _OutputIterator std::__gnu_parallel::set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator> _OutputIterator std::__parallel::set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate> _OutputIterator std::__gnu_parallel::set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator> _OutputIterator std::__parallel::set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate> _OutputIterator std::__gnu_parallel::set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator> _OutputIterator std::__parallel::set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate> _OutputIterator std::__gnu_parallel::set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator> _OutputIterator std::__parallel::set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate> _OutputIterator std::__gnu_parallel::set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator> _OutputIterator std::__parallel::set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate> _OutputIterator std::__gnu_parallel::set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _RAIter> void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare> void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism> void std::__parallel::sort (_RAIter __begin, \_\_gnu\_parallel::sequential\_tag, _Parallelism __parallelism)`
- `template<typename _RAIter> void std::__parallel::sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter> void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::default\_parallel\_tag, _Parallelism __parallelism)`
- `template<typename _RAIter> void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::parallel\_tag, _Parallelism __parallelism)`



- `template<typename _RAIter> void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway↵  
__mergesort_tag __parallelism)`
- `template<typename _RAIter> void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway↵  
__mergesort_sampling_tag __parallelism)`
- `template<typename _RAIter> void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway↵  
__mergesort_exact_tag __parallelism)`
- `template<typename _RAIter> void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_parallel↵  
::__quicksort_tag __parallelism)`
- `template<typename _RAIter> void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_parallel↵  
::__balanced_quicksort_tag __parallelism)`
- `template<typename _RAIter, typename _Compare> void std::__parallel::sort (_RAIter __begin, _RAIter __end, ↵  
_Compare __comp)`
- `template<typename _RAIter> void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel↵  
::__sequential_tag)`
- `template<typename _RAIter, typename _Compare> void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, ↵  
_Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism> void std::__parallel::stable_sort (_RAIter ↵  
__begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter> void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter> void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel↵  
::__default_parallel_tag __parallelism)`
- `template<typename _RAIter> void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel↵  
::__parallel_tag __parallelism)`
- `template<typename _RAIter> void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel↵  
::__multiway_mergesort_tag __parallelism)`
- `template<typename _RAIter> void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel↵  
::__quicksort_tag __parallelism)`
- `template<typename _RAIter> void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel↵  
::__balanced_quicksort_tag __parallelism)`
- `template<typename _RAIter, typename _Compare> void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, ↵  
_Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation> _OutputIterator std::__parallel↵  
::__transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op, __gnu↵  
__parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation> _OutputIterator std::__parallel::transform  
( _Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op, __gnu_parallel::↵  
Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation> _OutputIterator std::__parallel::transform  
( _Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation> _OutputIterator std::__↵  
__parallel::transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _Binary↵  
Operation __binary_op, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation> _OutputIterator std::__↵  
__parallel::transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _Binary↵  
Operation __binary_op, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation> _OutputIterator std::__↵  
__parallel::transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _Binary↵  
Operation __binary_op)`
- `template<typename _Iter, typename _OutputIterator> _OutputIterator std::__parallel::unique_copy (_Iter __begin1, ↵  
_Iter __end1, _OutputIterator __out, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate> _OutputIterator std::__parallel::unique_copy  
( _Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)`

- `template<typename _Iter , typename _OutputIterator > _OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out)`
- `template<typename _Iter , typename _OutputIterator , typename _Predicate > _OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate __pred)`

### 5.1.1 Detailed Description

Parallel STL function calls corresponding to the `stl_algo.h` header.

The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

## 5.2 `algo.h` File Reference

### Namespaces

- [std](#)
- [std::\\_\\_parallel](#)

### Functions

- `template<typename _Iter1 , typename _Iter2 , typename _Predicate , typename _IteratorTag1 , typename _IteratorTag2 > bool std::__parallel::lexicographical_compare_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1 , typename _RAIter2 , typename _Predicate > bool std::__parallel::lexicographical_compare_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1 , typename _Iter2 , typename _Predicate , typename _IteratorTag1 , typename _IteratorTag2 > pair< _Iter1, _Iter2 > std::__parallel::mismatch_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1 , typename _RAIter2 , typename _Predicate > pair< _RAIter1, _RAIter2 > std::__parallel::mismatch_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1 , typename _Iter2 > bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1 , typename _Iter2 , typename _Predicate > bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1 , typename _Iter2 > bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1 , typename _Iter2 , typename _Predicate > bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _Iter1 , typename _Iter2 > bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1 , typename _Iter2 , typename _Predicate > bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1 , typename _Iter2 > bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1 , typename _Iter2 , typename _Predicate > bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred)`

- `template<typename _Iter1, typename _Iter2> pair<_Iter1, _Iter2> std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate> pair<_Iter1, _Iter2> std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2> pair<_Iter1, _Iter2> std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate> pair<_Iter1, _Iter2> std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`

### 5.2.1 Detailed Description

Parallel STL function calls corresponding to the `stl_algobase.h` header. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

## 5.3 algorithmfwd.h File Reference

### Namespaces

- [std](#)

### Functions

- `template<typename _Filter> _Filter std::adjacent_find (_Filter, _Filter)`
- `template<typename _Filter, typename _BinaryPredicate> _Filter std::adjacent_find (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _Iter, typename _Predicate> bool std::all_of (_Iter, _Iter, _Predicate)`
- `template<typename _Iter, typename _Predicate> bool std::any_of (_Iter, _Iter, _Predicate)`
- `template<typename _Filter, typename _Tp> bool std::binary_search (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare> bool std::binary_search (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _Iter, typename _OIter> _OIter std::copy (_Iter, _Iter, _OIter)`
- `template<typename _BIter1, typename _BIter2> _BIter2 std::copy_backward (_BIter1, _BIter1, _BIter2)`
- `template<typename _Iter, typename _OIter, typename _Predicate> _OIter std::copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Iter, typename _Size, typename _OIter> _OIter std::copy_n (_Iter, _Size, _OIter)`
- `template<typename _Iter, typename _Tp> iterator_traits<_Iter>::difference_type std::count (_Iter, _Iter, const _Tp &)`
- `template<typename _Iter, typename _Predicate> iterator_traits<_Iter>::difference_type std::count_if (_Iter, _Iter, _Predicate)`
- `template<typename _Iter1, typename _Iter2> bool std::equal (_Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate> bool std::equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _Filter, typename _Tp> pair<_Filter, _Filter> std::equal_range (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare> pair<_Filter, _Filter> std::equal_range (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _Filter, typename _Tp> void std::fill (_Filter, _Filter, const _Tp &)`
- `template<typename _OIter, typename _Size, typename _Tp> _OIter std::fill_n (_OIter, _Size, const _Tp &)`
- `template<typename _Iter, typename _Tp> _Iter std::find (_Iter, _Iter, const _Tp &)`
- `template<typename _Filter1, typename _Filter2> _Filter1 std::find_end (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate> _Filter1 std::find_end (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`



- `template<typename _Filter1, typename _Filter2 > _Filter1 std::find_first_of (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate > _Filter1 std::find_first_of (_Filter1, _Filter1, _↵  
_Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _Iter, typename _Predicate > _Iter std::find_if (_Iter, _Iter, _Predicate)`
- `template<typename _Iter, typename _Predicate > _Iter std::find_if_not (_Iter, _Iter, _Predicate)`
- `template<typename _Iter, typename _Funct > _Funct std::for_each (_Iter, _Iter, _Funct)`
- `template<typename _Filter, typename _Generator > void std::generate (_Filter, _Filter, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator > _OIter std::generate_n (_OIter, _Size, _Generator)`
- `template<typename _Iter1, typename _Iter2 > bool std::includes (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare > bool std::includes (_Iter1, _Iter1, _Iter2, _Iter2,  
_Compare)`
- `template<typename _BIter > void std::inplace_merge (_BIter, _BIter, _BIter)`
- `template<typename _BIter, typename _Compare > void std::inplace_merge (_BIter, _BIter, _BIter, _Compare)`
- `template<typename _RAIter > bool std::is_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare > bool std::is_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter > _RAIter std::is_heap_until (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare > _RAIter std::is_heap_until (_RAIter, _RAIter, _Compare)`
- `template<typename _Iter, typename _Predicate > bool std::is_partitioned (_Iter, _Iter, _Predicate)`
- `template<typename _Filter1, typename _Filter2 > bool std::is_permutation (_Filter1, _Filter1, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate > bool std::is_permutation (_Filter1, _Filter1, _↵  
_Filter2, _BinaryPredicate)`
- `template<typename _Filter > bool std::is_sorted (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare > bool std::is_sorted (_Filter, _Filter, _Compare)`
- `template<typename _Filter > _Filter std::is_sorted_until (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare > _Filter std::is_sorted_until (_Filter, _Filter, _Compare)`
- `template<typename _Filter1, typename _Filter2 > void std::iter_swap (_Filter1, _Filter2)`
- `template<typename _Iter1, typename _Iter2 > bool std::lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare > bool std::lexicographical_compare (_Iter1, _Iter1,  
_Iter2, _Iter2, _Compare)`
- `template<typename _Filter, typename _Tp > _Filter std::lower_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare > _Filter std::lower_bound (_Filter, _Filter, const _Tp &,  
_Compare)`
- `template<typename _RAIter > void std::make_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare > void std::make_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _Tp > const _Tp & std::max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare > const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare  
__comp)`
- `template<typename _Tp > _Tp std::max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare > _Tp std::max (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter > _Filter std::max_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare > _Filter std::max_element (_Filter, _Filter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter > _OIter std::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare > _OIter std::merge (_Iter1, _Iter1,  
_Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Tp > const _Tp & std::min (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare > const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare  
__comp)`
- `template<typename _Tp > _Tp std::min (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare > _Tp std::min (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter > _Filter std::min_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare > _Filter std::min_element (_Filter, _Filter, _Compare)`

- `template<typename _Tp> pair< const _Tp &, const _Tp &> std::minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare> pair< const _Tp &, const _Tp &> std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp> pair< _Tp, _Tp> std::minmax (initializer_list< _Tp>)`
- `template<typename _Tp, typename _Compare> pair< _Tp, _Tp> std::minmax (initializer_list< _Tp>, _Compare)`
- `template<typename _Filter> pair< _Filter, _Filter> std::minmax_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare> pair< _Filter, _Filter> std::minmax_element (_Filter, _Filter, _Compare)`
- `template<typename _Ilter1, typename _Ilter2> pair< _Ilter1, _Ilter2> std::mismatch (_Ilter1, _Ilter1, _Ilter2)`
- `template<typename _Ilter1, typename _Ilter2, typename _BinaryPredicate> pair< _Ilter1, _Ilter2> std::mismatch (_Ilter1, _Ilter1, _Ilter2, _BinaryPredicate)`
- `template<typename _Blter> bool std::next_permutation (_Blter, _Blter)`
- `template<typename _Blter, typename _Compare> bool std::next_permutation (_Blter, _Blter, _Compare)`
- `template<typename _Ilter, typename _Predicate> bool std::none_of (_Ilter, _Ilter, _Predicate)`
- `template<typename _RAlter> void std::nth_element (_RAlter, _RAlter, _RAlter)`
- `template<typename _RAlter, typename _Compare> void std::nth_element (_RAlter, _RAlter, _RAlter, _Compare)`
- `template<typename _RAlter> void std::partial_sort (_RAlter, _RAlter, _RAlter)`
- `template<typename _RAlter, typename _Compare> void std::partial_sort (_RAlter, _RAlter, _RAlter, _Compare)`
- `template<typename _Ilter, typename _RAlter> _RAlter std::partial_sort_copy (_Ilter, _Ilter, _RAlter, _RAlter)`
- `template<typename _Ilter, typename _RAlter, typename _Compare> _RAlter std::partial_sort_copy (_Ilter, _Ilter, _RAlter, _RAlter, _Compare)`
- `template<typename _Blter, typename _Predicate> _Blter std::partition (_Blter, _Blter, _Predicate)`
- `template<typename _Ilter, typename _Olter1, typename _Olter2, typename _Predicate> pair< _Olter1, _Olter2> std::partition_copy (_Ilter, _Ilter, _Olter1, _Olter2, _Predicate)`
- `template<typename _Filter, typename _Predicate> _Filter std::partition_point (_Filter, _Filter, _Predicate)`
- `template<typename _RAlter> void std::pop_heap (_RAlter, _RAlter)`
- `template<typename _RAlter, typename _Compare> void std::pop_heap (_RAlter, _RAlter, _Compare)`
- `template<typename _Blter> bool std::prev_permutation (_Blter, _Blter)`
- `template<typename _Blter, typename _Compare> bool std::prev_permutation (_Blter, _Blter, _Compare)`
- `template<typename _RAlter> void std::push_heap (_RAlter, _RAlter)`
- `template<typename _RAlter, typename _Compare> void std::push_heap (_RAlter, _RAlter, _Compare)`
- `template<typename _RAlter> void std::random_shuffle (_RAlter, _RAlter)`
- `template<typename _RAlter, typename _Generator> void std::random_shuffle (_RAlter, _RAlter, _Generator &&)`
- `template<typename _Filter, typename _Tp> _Filter std::remove (_Filter, _Filter, const _Tp &)`
- `template<typename _Ilter, typename _Olter, typename _Tp> _Olter std::remove_copy (_Ilter, _Ilter, _Olter, const _Tp &)`
- `template<typename _Ilter, typename _Olter, typename _Predicate> _Olter std::remove_copy_if (_Ilter, _Ilter, _Olter, _Predicate)`
- `template<typename _Filter, typename _Predicate> _Filter std::remove_if (_Filter, _Filter, _Predicate)`
- `template<typename _Filter, typename _Tp> void std::replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _Ilter, typename _Olter, typename _Tp> _Olter std::replace_copy (_Ilter, _Ilter, _Olter, const _Tp &, const _Tp &)`
- `template<typename _Ilter, typename _Olter, typename _Predicate, typename _Tp> _Olter std::replace_copy_if (_Ilter, _Ilter, _Olter, _Predicate, const _Tp &)`
- `template<typename _Filter, typename _Predicate, typename _Tp> void std::replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _Blter> void std::reverse (_Blter, _Blter)`
- `template<typename _Blter, typename _Olter> _Olter std::reverse_copy (_Blter, _Blter, _Olter)`
- `template<typename _Filter> void std::rotate (_Filter, _Filter, _Filter)`
- `template<typename _Filter, typename _Olter> _Olter std::rotate_copy (_Filter, _Filter, _Filter, _Olter)`
- `template<typename _Filter1, typename _Filter2> _Filter1 std::search (_Filter1, _Filter1, _Filter2, _Filter2)`

- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate > _Filter1 std::search (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _Filter, typename _Size, typename _Tp > _Filter std::search_n (_Filter, _Filter, _Size, const _Tp &)`
- `template<typename _Filter, typename _Size, typename _Tp, typename _BinaryPredicate > _Filter std::search_n (_Filter, _Filter, _Size, const _Tp &, _BinaryPredicate)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter > _Olter std::set_difference (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter, typename _Compare > _Olter std::set_difference (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter, _Compare)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter > _Olter std::set_intersection (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter, typename _Compare > _Olter std::set_intersection (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter, _Compare)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter > _Olter std::set_symmetric_difference (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter, typename _Compare > _Olter std::set_symmetric_difference (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter, _Compare)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter > _Olter std::set_union (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter, typename _Compare > _Olter std::set_union (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter, _Compare)`
- `template<typename _RAlter, typename _UGenerator > void std::shuffle (_RAlter, _RAlter, _UGenerator &&)`
- `template<typename _RAlter > void std::sort (_RAlter, _RAlter)`
- `template<typename _RAlter, typename _Compare > void std::sort (_RAlter, _RAlter, _Compare)`
- `template<typename _RAlter > void std::sort_heap (_RAlter, _RAlter)`
- `template<typename _RAlter, typename _Compare > void std::sort_heap (_RAlter, _RAlter, _Compare)`
- `template<typename _Blter, typename _Predicate > _Blter std::stable_partition (_Blter, _Blter, _Predicate)`
- `template<typename _RAlter > void std::stable_sort (_RAlter, _RAlter)`
- `template<typename _RAlter, typename _Compare > void std::stable_sort (_RAlter, _RAlter, _Compare)`
- `template<typename _Tp > void std::swap (_Tp &__a, _Tp &__b) noexcept(__and< is_nothrow_move_constructible< _Tp >, is_nothrow_move_assignable< _Tp >>::value)`
- `template<typename _Tp, size_t _Nm> void std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(noexcept(swap(*__a, *__b)))`
- `template<typename _Filter1, typename _Filter2 > _Filter2 std::swap_ranges (_Filter1, _Filter1, _Filter2)`
- `template<typename _Ilter, typename _Olter, typename _UnaryOperation > _Olter std::transform (_Ilter, _Ilter, _Olter, _UnaryOperation)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter, typename _BinaryOperation > _Olter std::transform (_Ilter1, _Ilter1, _Ilter2, _Olter, _BinaryOperation)`
- `template<typename _Filter > _Filter std::unique (_Filter, _Filter)`
- `template<typename _Filter, typename _BinaryPredicate > _Filter std::unique (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _Ilter, typename _Olter > _Olter std::unique_copy (_Ilter, _Ilter, _Olter)`
- `template<typename _Ilter, typename _Olter, typename _BinaryPredicate > _Olter std::unique_copy (_Ilter, _Ilter, _Olter, _BinaryPredicate)`
- `template<typename _Filter, typename _Tp > _Filter std::upper_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare > _Filter std::upper_bound (_Filter, _Filter, const _Tp &, _Compare)`

### 5.3.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

## 5.4 algorithmfwd.h File Reference

### Namespaces

- [std](#)
- [std::\\_\\_parallel](#)

### Functions

- `template<typename _Filter, typename _IterTag> _Filter std::__parallel::__adjacent_find_switch (_Filter, _Filter, _IterTag)`
- `template<typename _Filter, typename _BiPredicate, typename _IterTag> _Filter std::__parallel::__adjacent_find_switch (_Filter, _Filter, _BiPredicate, _IterTag)`
- `template<typename _RAIter, typename _BiPredicate> _RAIter std::__parallel::__adjacent_find_switch (_RAIter, _RAIter, _BiPredicate, random_access_iterator_tag)`
- `template<typename _RAIter> _RAIter std::__parallel::__adjacent_find_switch (_RAIter __begin, _RAIter __end, random_access_iterator_tag)`
- `template<typename _Iter, typename _Predicate, typename _IterTag> iterator_traits<_Iter>::difference_type std::__parallel::__count_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate> iterator_traits<_RAIter>::difference_type std::__parallel::__count_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_unbalanced)`
- `template<typename _Iter, typename _Tp, typename _IterTag> iterator_traits<_Iter>::difference_type std::__parallel::__count_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp> iterator_traits<_RAIter>::difference_type std::__parallel::__count_switch (_RAIter __begin, _RAIter __end, const _Tp & __value, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_unbalanced)`
- `template<typename _Iter, typename _Filter, typename _IterTag1, typename _IterTag2> _Iter std::__parallel::__find_first_of_switch (_Iter, _Iter, _Filter, _Filter, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _Filter, typename _BiPredicate, typename _IterTag> _RAIter std::__parallel::__find_first_of_switch (_RAIter, _RAIter, _Filter, _Filter, _BiPredicate, random_access_iterator_tag, _IterTag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate, typename _IterTag1, typename _IterTag2> _Iter std::__parallel::__find_first_of_switch (_Iter, _Iter, _Filter, _Filter, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _Iter, typename _Predicate, typename _IterTag> _Iter std::__parallel::__find_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate> _RAIter std::__parallel::__find_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _RAIter, typename _Tp> _RAIter std::__parallel::__find_switch (_RAIter __begin, _RAIter __end, const _Tp & __val, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _IterTag> _Iter std::__parallel::__find_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Function> _Function std::__parallel::__for_each_switch (_RAIter __begin, _RAIter __end, _Function __f, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _Iter, typename _Function, typename _IterTag> _Function std::__parallel::__for_each_switch (_Iter, _Iter, _Function, _IterTag)`
- `template<typename _OIter, typename _Size, typename _Generator, typename _IterTag> _OIter std::__parallel::__generate_n_switch (_OIter, _Size, _Generator, _IterTag)`
- `template<typename _RAIter, typename _Size, typename _Generator> _RAIter std::__parallel::__generate_n_switch (_RAIter __begin, _Size __n, _Generator __gen, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`

- `template<typename _Filter, typename _Generator, typename _IterTag> void std::parallel::generate_switch (_Filter, _Filter, _Generator, _IterTag)`
- `template<typename _RAIter, typename _Generator> void std::parallel::generate_switch (_RAIter __begin, _RAIter __end, _Generator __gen, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag=gnu\_parallel::parallel\_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate> bool std::parallel::lexicographical_compare_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2> bool std::parallel::lexicographical_compare_switch (_Iter1, _Iter1, _Iter2, _Iter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _Filter, typename _Compare, typename _IterTag> _Filter std::parallel::max_element_switch (_Filter, _Filter, _Compare, _IterTag)`
- `template<typename _RAIter, typename _Compare> _RAIter std::parallel::max_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag=gnu\_parallel::parallel\_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare, typename _IterTag1, typename _IterTag2, typename _IterTag3> _OIter std::parallel::merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare> _OIter std::parallel::merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter, typename _Compare, typename _IterTag> _Filter std::parallel::min_element_switch (_Filter, _Filter, _Compare, _IterTag)`
- `template<typename _RAIter, typename _Compare> _RAIter std::parallel::min_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag=gnu\_parallel::parallel\_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate> pair<_RAIter1, _RAIter2> std::parallel::mismatch_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2> pair<_Iter1, _Iter2> std::parallel::mismatch_switch (_Iter1, _Iter1, _Iter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _Filter, typename _Predicate, typename _IterTag> _Filter std::parallel::partition_switch (_Filter, _Filter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate> _RAIter std::parallel::partition_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp, typename _IterTag> void std::parallel::replace_if_switch (_Filter, _Filter, _Predicate, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Predicate, typename _Tp> void std::parallel::replace_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp & __new_value, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag=gnu\_parallel::parallel\_balanced)`
- `template<typename _Filter, typename _Tp, typename _IterTag> void std::parallel::replace_switch (_Filter, _Filter, const _Tp &, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp> void std::parallel::replace_switch (_RAIter __begin, _RAIter __end, const _Tp & __old_value, const _Tp & __new_value, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag=gnu\_parallel::parallel\_balanced)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BiPredicate> _RAIter std::parallel::search_n_switch (_RAIter, _RAIter, _Integer, const _Tp &, _BiPredicate, random_access_iterator_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate, typename _IterTag> _Filter std::parallel::search_n_switch (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, _IterTag)`
- `template<typename _Filter1, typename _Filter2, typename _IterTag1, typename _IterTag2> _Filter1 std::parallel::search_switch (_Filter1, _Filter1, _Filter2, _Filter2, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BiPredicate> _RAIter1 std::parallel::search_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter2, _BiPredicate, random_access_iterator_tag, random_access_iterator_tag)`

- `template<typename _Filter1, typename _Filter2, typename _BiPredicate, typename _IterTag1, typename _IterTag2 > _Filter1 std::↵  
_parallel:: search_switch ( _Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2 > _RAIter1 std:: _parallel:: search_switch ( _RAIter1 __begin1, ↵  
_RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, random_access_iterator_tag, random_access_iterator↵  
_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate > _Output_RAlter std::↵  
_parallel:: set_difference_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 ↵  
__end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag,  
random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2 ,  
typename _IterTag3 > _OIter std:: _parallel:: set_difference_switch ( _Iter1, _Iter1, _Iter2, _Iter2, _OIter, ↵  
_Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate > _Output_RAlter std::↵  
_parallel:: set_intersection_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 ↵  
__end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag,  
random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2 ,  
typename _IterTag3 > _OIter std:: _parallel:: set_intersection_switch ( _Iter1, _Iter1, _Iter2, _Iter2, _OIter, ↵  
_Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate > _Output_RAlter std::↵  
_parallel:: set_symmetric_difference_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2,  
_RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access↵  
_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2 ,  
typename _IterTag3 > _OIter std:: _parallel:: set_symmetric_difference_switch ( _Iter1, _Iter1, _Iter2, _Iter2, ↵  
_OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate > _Output_RAlter std↵  
:: _parallel:: set_union_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 ↵  
__end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag,  
random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2 ,  
typename _IterTag3 > _OIter std:: _parallel:: set_union_switch ( _Iter1, _Iter1, _Iter2, _Iter2, _OIter, ↵  
_Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation, typename _IterTag1, typename _IterTag2 > _OIter std::↵  
_parallel:: transform1_switch ( _Iter, _Iter, _OIter, _UnaryOperation, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RAOIter, typename _UnaryOperation > _RAOIter std:: _parallel:: transform1↵  
switch ( _RAIter, _RAIter, _RAOIter, _UnaryOperation, random_access_iterator_tag, random_access_iterator↵  
_tag, \_\_gnu\_parallel::Parallelism __parallelism=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BiOperation > _RAIter3 std:: _parallel↵  
:: transform2_switch ( _RAIter1, _RAIter1, _RAIter2, _RAIter3, _BiOperation, random_access_iterator_tag,  
random_access_iterator_tag, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism=\_\_gnu↵  
\_parallel::parallel\_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation, typename _Tag1, typename _Tag2, typename  
_Tag3 > _OIter std:: _parallel:: transform2_switch ( _Iter1, _Iter1, _Iter2, _OIter, _BiOperation, _Tag1, ↵  
_Tag2, _Tag3)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _IterTag1, typename _IterTag2 > _OIter std::↵  
_parallel:: unique_copy_switch ( _Iter, _Iter, _OIter, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RandomAccess_OIter, typename _Predicate > _RandomAccess_OIter std::↵  
_parallel:: unique_copy_switch ( _RAIter, _RAIter, _RandomAccess_OIter, _Predicate, random_access↵  
_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter > _Filter std:: _parallel::adjacent_find ( _Filter, _Filter)`
- `template<typename _Filter > _Filter std:: _parallel::adjacent_find ( _Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _BiPredicate > _Filter std:: _parallel::adjacent_find ( _Filter, _Filter, _BiPredicate)`



- `template<typename _Filter, typename _BiPredicate> _Filter std::__parallel::adjacent_find (_Filter, _Filter, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp> iterator_traits<_Iter>::difference_type std::__parallel::count (_Iter __begin, _Iter __end, const _Tp &__value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp> iterator_traits<_Iter>::difference_type std::__parallel::count (_Iter __begin, _Iter __end, const _Tp &__value, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp> iterator_traits<_Iter>::difference_type std::__parallel::count (_Iter __begin, _Iter __end, const _Tp &__value)`
- `template<typename _Iter, typename _Predicate> iterator_traits<_Iter>::difference_type std::__parallel::count_if (<_Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate> iterator_traits<_Iter>::difference_type std::__parallel::count_if (<_Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate> iterator_traits<_Iter>::difference_type std::__parallel::count_if (<_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2> bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate> bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2> bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate> bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _Iter, typename _Tp> _Iter std::__parallel::find (_Iter __begin, _Iter __end, const _Tp &__val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp> _Iter std::__parallel::find (_Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _Filter> _Iter std::__parallel::find_first_of (_Iter, _Iter, _Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate> _Iter std::__parallel::find_first_of (_Iter, _Iter, <_Filter, _Filter, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate> _Iter std::__parallel::find_first_of (_Iter, _Iter, <_Filter, _Filter, _BiPredicate)`
- `template<typename _Iter, typename _Filter> _Iter std::__parallel::find_first_of (_Iter, _Iter, _Filter, _Filter)`
- `template<typename _Iter, typename _Predicate> _Iter std::__parallel::find_if (_Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate> _Iter std::__parallel::find_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Function> _Function std::__parallel::for_each (_Iter __begin, _Iter __end, <_Function __f, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iterator, typename _Function> _Function std::__parallel::for_each (_Iterator __begin, _Iterator __end, _Function __f, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Function> _Function std::__parallel::for_each (_Iter, _Iter, _Function)`
- `template<typename _Filter, typename _Generator> void std::__parallel::generate (_Filter, _Filter, _Generator)`
- `template<typename _Filter, typename _Generator> void std::__parallel::generate (_Filter, _Filter, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Generator> void std::__parallel::generate (_Filter, _Filter, _Generator, \_\_gnu\_parallel::Parallelism)`
- `template<typename _OIter, typename _Size, typename _Generator> _OIter std::__parallel::generate_n (_OIter, _Size, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator> _OIter std::__parallel::generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OIter, typename _Size, typename _Generator> _OIter std::__parallel::generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::Parallelism)`

- `template<typename _Iter1, typename _Iter2> bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate> bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2> bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate> bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred)`
- `template<typename _Filter> _Filter std::__parallel::max_element (_Filter, _Filter)`
- `template<typename _Filter> _Filter std::__parallel::max_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter> _Filter std::__parallel::max_element (_Filter, _Filter, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Compare> _Filter std::__parallel::max_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare> _Filter std::__parallel::max_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Compare> _Filter std::__parallel::max_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _OIter> _OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare> _OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare> _OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter> _OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Filter> _Filter std::__parallel::min_element (_Filter, _Filter)`
- `template<typename _Filter> _Filter std::__parallel::min_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter> _Filter std::__parallel::min_element (_Filter, _Filter, \_\_gnu\_parallel::Parallelism, \_\_gnu\_parallel::parallelism\_tag)`
- `template<typename _Filter, typename _Compare> _Filter std::__parallel::min_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare> _Filter std::__parallel::min_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Compare> _Filter std::__parallel::min_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2> pair<_Iter1, _Iter2> std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate> pair<_Iter1, _Iter2> std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2> pair<_Iter1, _Iter2> std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate> pair<_Iter1, _Iter2> std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _RAIter> void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare> void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare> void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter> void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end)`
- `template<typename _RAIter, typename _Compare> void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`



- `template<typename _RAIter> void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare> void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter> void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end)`
- `template<typename _Filter, typename _Predicate> _Filter std::__parallel::partition (_Filter, _Filter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Predicate> _Filter std::__parallel::partition (_Filter, _Filter, _Predicate)`
- `template<typename _RAIter> void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _RandomNumberGenerator> void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator & __rand, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter> void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter, typename _RandomNumberGenerator> void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator && __rand)`
- `template<typename _Filter, typename _Tp> void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _Filter, typename _Tp> void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Tp> void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Predicate, typename _Tp> void std::__parallel::replace_if (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag, _Predicate, const _Tp &)`
- `template<typename _Filter, typename _Predicate, typename _Tp> void std::__parallel::replace_if (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag, _Predicate, const _Tp &, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter1, typename _Filter2> _Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter1, typename _Filter2> _Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate> _Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate> _Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate)`
- `template<typename _Filter, typename _Integer, typename _Tp> _Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate> _Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp> _Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate> _Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter> _OIter std::__parallel::set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate> _OIter std::__parallel::set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter> _OIter std::__parallel::set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate> _OIter std::__parallel::set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter> _OIter std::__parallel::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate> _OIter std::__parallel::set_intersection(_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter> _OIter std::__parallel::set_intersection(_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate> _OIter std::__parallel::set_intersection(_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter> _OIter std::__parallel::set_symmetric_difference(_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate> _OIter std::__parallel::set_symmetric_difference(_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter> _OIter std::__parallel::set_symmetric_difference(_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate> _OIter std::__parallel::set_symmetric_difference(_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter> _OIter std::__parallel::set_union(_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate> _OIter std::__parallel::set_union(_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter> _OIter std::__parallel::set_union(_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate> _OIter std::__parallel::set_union(_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _RAIter> void std::__parallel::sort(_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare> void std::__parallel::sort(_RAIter __begin, _RAIter __end, __Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter> void std::__parallel::sort(_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter, typename _Compare> void std::__parallel::sort(_RAIter __begin, _RAIter __end, __Compare __comp)`
- `template<typename _RAIter> void std::__parallel::stable_sort(_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare> void std::__parallel::stable_sort(_RAIter __begin, _RAIter __end, __Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter> void std::__parallel::stable_sort(_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter, typename _Compare> void std::__parallel::stable_sort(_RAIter __begin, _RAIter __end, __Compare __comp)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation> _OIter std::__parallel::transform(_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation> _OIter std::__parallel::transform(_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation> _OIter std::__parallel::transform(_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation> _OIter std::__parallel::transform(_Iter1, _Iter1, _Iter2, _OIter, _BiOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation> _OIter std::__parallel::transform(_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation> _OIter std::__parallel::transform(_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter> _OIter std::__parallel::unique_copy(_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _Predicate> _OIter std::__parallel::unique_copy(_Iter, _Iter, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Iter, typename _OIter> _OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _Predicate> _OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, _Predicate)`

#### 5.4.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

### 5.5 alloc\_traits.h File Reference

#### Classes

- struct `std::allocator\_traits<_Alloc>`

#### Namespaces

- `std`

#### Macros

- `#define _GLIBCXX_ALLOC_TR_NESTED_TYPE(_NTYPE, _ALT)`

#### Typedefs

- `template<typename _Alloc> using std::__check_copy_constructible = __allow_copy_cons<__is_copy_constructible<_Alloc>::value>`

#### Functions

- `template<typename _Alloc> void std::__alloc_on_copy (_Alloc &__one, const _Alloc &__two)`
- `template<typename _Alloc> _Alloc std::__alloc_on_copy (const _Alloc &__a)`
- `template<typename _Alloc> void std::__alloc_on_move (_Alloc &__one, _Alloc &__two)`
- `template<typename _Alloc> void std::__alloc_on_swap (_Alloc &__one, _Alloc &__two)`
- `template<typename _Alloc> void std::__do_alloc_on_copy (_Alloc &__one, const _Alloc &__two, true_type)`
- `template<typename _Alloc> void std::__do_alloc_on_copy (_Alloc &, const _Alloc &, false_type)`
- `template<typename _Alloc> void std::__do_alloc_on_move (_Alloc &__one, _Alloc &__two, true_type)`
- `template<typename _Alloc> void std::__do_alloc_on_move (_Alloc &, _Alloc &, false_type)`
- `template<typename _Alloc> void std::__do_alloc_on_swap (_Alloc &__one, _Alloc &__two, true_type)`
- `template<typename _Alloc> void std::__do_alloc_on_swap (_Alloc &, _Alloc &, false_type)`

#### 5.5.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 5.6 alloc\_traits.h File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits<\\_Alloc>](#)
- class [\\_\\_gnu\\_cxx::array\\_allocator<\\_Tp, \\_Array>](#)
- class [\\_\\_gnu\\_cxx::bitmap\\_allocator<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::malloc\\_allocator<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::new\\_allocator<\\_Tp>](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)

#### 5.6.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.7 allocator.h File Reference

### Classes

- class [std::allocator<\\_Tp>](#)
- class [std::allocator<void>](#)

### Namespaces

- [std](#)

### Functions

- template<typename \_T1, typename \_T2> bool **std::operator!=** (const allocator<\_T1> &, const allocator<\_T2> &)
- template<typename \_Tp> bool **std::operator!=** (const allocator<\_Tp> &, const allocator<\_Tp> &)
- template<typename \_T1, typename \_T2> bool **std::operator==** (const allocator<\_T1> &, const allocator<\_T2> &)
- template<typename \_Tp> bool **std::operator==** (const allocator<\_Tp> &, const allocator<\_Tp> &)

#### 5.7.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 5.8 array\_allocator.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::array\\_allocator<\\_Tp, \\_Array>](#)
- class [\\_\\_gnu\\_cxx::array\\_allocator\\_base<\\_Tp>](#)

## Namespaces

- [\\_\\_gnu\\_cxx](#)

## Functions

- `template<typename _Tp, typename _Array> bool \_\_gnu\_cxx::operator!= (const array_allocator< _Tp, _Array > &, const array_allocator< _Tp, _Array > &)`
- `template<typename _Tp, typename _Array> bool \_\_gnu\_cxx::operator== (const array_allocator< _Tp, _Array > &, const array_allocator< _Tp, _Array > &)`

## 5.8.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

5.9 `assoc_container.hpp` File Reference

## Classes

- class [\\_\\_gnu\\_pbds::basic\\_branch](#)< Key, Mapped, Tag, Node\_Update, Policy\_Tl, \_Alloc >
- class [\\_\\_gnu\\_pbds::basic\\_hash\\_table](#)< Key, Mapped, Hash\_Fn, Eq\_Fn, Resize\_Policy, Store\_Hash, Tag, Policy\_Tl, \_Alloc >
- class [\\_\\_gnu\\_pbds::cc\\_hash\\_table](#)< Key, Mapped, Hash\_Fn, Eq\_Fn, Comb\_Hash\_Fn, Resize\_Policy, Store\_Hash, \_Alloc >
- class [\\_\\_gnu\\_pbds::gp\\_hash\\_table](#)< Key, Mapped, Hash\_Fn, Eq\_Fn, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy, Store\_Hash, \_Alloc >
- class [\\_\\_gnu\\_pbds::list\\_update](#)< Key, Mapped, Eq\_Fn, Update\_Policy, \_Alloc >
- class [\\_\\_gnu\\_pbds::tree](#)< Key, Mapped, Cmp\_Fn, Tag, Node\_Update, \_Alloc >
- class [\\_\\_gnu\\_pbds::trie](#)< Key, Mapped, \_ATraits, Tag, Node\_Update, \_Alloc >

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB\_DS\_BRANCH\_BASE`
- `#define PB\_DS\_CC\_HASH\_BASE`
- `#define PB\_DS\_GP\_HASH\_BASE`
- `#define PB\_DS\_HASH\_BASE`
- `#define PB\_DS\_LU\_BASE`
- `#define PB\_DS\_TREE\_BASE`
- `#define PB\_DS\_TREE\_NODE\_AND\_IT\_TRAITS`
- `#define PB\_DS\_TRIE\_BASE`
- `#define PB\_DS\_TRIE\_NODE\_AND\_IT\_TRAITS`

## 5.9.1 Detailed Description

Contains associative containers.

## 5.10 atomic\_base.h File Reference

### Classes

- struct [std::\\_\\_atomic\\_base< \\_ITp >](#)
- struct [std::\\_\\_atomic\\_base< \\_ITp >](#)
- struct [std::\\_\\_atomic\\_base< \\_PTp \\* >](#)
- struct [std::\\_\\_atomic\\_flag\\_base](#)
- struct [std::atomic\\_flag](#)

### Namespaces

- [std](#)

### Macros

- `#define ATOMIC_FLAG_INIT`
- `#define ATOMIC_VAR_INIT(_VI)`

### Typedefs

- typedef unsigned char [std::\\_\\_atomic\\_flag\\_data\\_type](#)
- typedef \_\_atomic\_base< char > [std::atomic\\_char](#)
- typedef \_\_atomic\_base< char16\_t > [std::atomic\\_char16\\_t](#)
- typedef \_\_atomic\_base< char32\_t > [std::atomic\\_char32\\_t](#)
- typedef \_\_atomic\_base< int > [std::atomic\\_int](#)
- typedef \_\_atomic\_base< int\_fast16\_t > [std::atomic\\_int\\_fast16\\_t](#)
- typedef \_\_atomic\_base< int\_fast32\_t > [std::atomic\\_int\\_fast32\\_t](#)
- typedef \_\_atomic\_base< int\_fast64\_t > [std::atomic\\_int\\_fast64\\_t](#)
- typedef \_\_atomic\_base< int\_fast8\_t > [std::atomic\\_int\\_fast8\\_t](#)
- typedef \_\_atomic\_base< int\_least16\_t > [std::atomic\\_int\\_least16\\_t](#)
- typedef \_\_atomic\_base< int\_least32\_t > [std::atomic\\_int\\_least32\\_t](#)
- typedef \_\_atomic\_base< int\_least64\_t > [std::atomic\\_int\\_least64\\_t](#)
- typedef \_\_atomic\_base< int\_least8\_t > [std::atomic\\_int\\_least8\\_t](#)
- typedef \_\_atomic\_base< intmax\_t > [std::atomic\\_intmax\\_t](#)
- typedef \_\_atomic\_base< intptr\_t > [std::atomic\\_intptr\\_t](#)
- typedef \_\_atomic\_base< long long > [std::atomic\\_llong](#)
- typedef \_\_atomic\_base< long > [std::atomic\\_long](#)
- typedef \_\_atomic\_base< ptrdiff\_t > [std::atomic\\_ptrdiff\\_t](#)
- typedef \_\_atomic\_base< signed char > [std::atomic\\_schar](#)
- typedef \_\_atomic\_base< short > [std::atomic\\_short](#)
- typedef \_\_atomic\_base< size\_t > [std::atomic\\_size\\_t](#)
- typedef \_\_atomic\_base< unsigned char > [std::atomic\\_uchar](#)
- typedef \_\_atomic\_base< unsigned int > [std::atomic\\_uint](#)
- typedef \_\_atomic\_base< uint\_fast16\_t > [std::atomic\\_uint\\_fast16\\_t](#)
- typedef \_\_atomic\_base< uint\_fast32\_t > [std::atomic\\_uint\\_fast32\\_t](#)
- typedef \_\_atomic\_base< uint\_fast64\_t > [std::atomic\\_uint\\_fast64\\_t](#)
- typedef \_\_atomic\_base< uint\_fast8\_t > [std::atomic\\_uint\\_fast8\\_t](#)
- typedef \_\_atomic\_base< uint\_least16\_t > [std::atomic\\_uint\\_least16\\_t](#)
- typedef \_\_atomic\_base< uint\_least32\_t > [std::atomic\\_uint\\_least32\\_t](#)

- typedef \_\_atomic\_base< uint\_least64\_t > [std::atomic\\_uint\\_least64\\_t](#)
- typedef \_\_atomic\_base< uint\_least8\_t > [std::atomic\\_uint\\_least8\\_t](#)
- typedef \_\_atomic\_base< uintmax\_t > [std::atomic\\_uintmax\\_t](#)
- typedef \_\_atomic\_base< uintptr\_t > [std::atomic\\_uintptr\\_t](#)
- typedef \_\_atomic\_base< unsigned long long > [std::atomic\\_ullong](#)
- typedef \_\_atomic\_base< unsigned long > [std::atomic\\_ulong](#)
- typedef \_\_atomic\_base< unsigned short > [std::atomic\\_ushort](#)
- typedef \_\_atomic\_base< wchar\_t > [std::atomic\\_wchar\\_t](#)
- typedef enum [std::memory\\_order](#) [std::memory\\_order](#)

## Enumerations

- enum [\\_\\_memory\\_order\\_modifier](#) { [\\_\\_memory\\_order\\_mask](#), [\\_\\_memory\\_order\\_modifier\\_mask](#), [\\_\\_memory\\_order\\_hle\\_acquire](#), [\\_\\_memory\\_order\\_hle\\_release](#) }
- enum [std::memory\\_order](#) { [memory\\_order\\_relaxed](#), [memory\\_order\\_consume](#), [memory\\_order\\_acquire](#), [memory\\_order\\_release](#), [memory\\_order\\_acq\\_rel](#), [memory\\_order\\_seq\\_cst](#) }

## Functions

- constexpr memory\_order [std::\\_\\_cmpexch\\_failure\\_order](#) (memory\_order \_\_m) noexcept
- constexpr memory\_order [std::\\_\\_cmpexch\\_failure\\_order2](#) (memory\_order \_\_m) noexcept
- void [std::atomic\\_signal\\_fence](#) (memory\_order \_\_m) noexcept
- void [std::atomic\\_thread\\_fence](#) (memory\_order \_\_m) noexcept
- template<typename \_Tp > \_Tp [std::kill\\_dependency](#) (\_Tp \_\_y) noexcept
- constexpr memory\_order [std::operator&](#) (memory\_order \_\_m, \_\_memory\_order\_modifier \_\_mod)
- constexpr memory\_order [std::operator|](#) (memory\_order \_\_m, \_\_memory\_order\_modifier \_\_mod)

### 5.10.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<atomic>`.

## 5.11 `atomic_lockfree_defines.h` File Reference

### Macros

- #define [ATOMIC\\_BOOL\\_LOCK\\_FREE](#)
- #define [ATOMIC\\_CHAR16\\_T\\_LOCK\\_FREE](#)
- #define [ATOMIC\\_CHAR32\\_T\\_LOCK\\_FREE](#)
- #define [ATOMIC\\_CHAR\\_LOCK\\_FREE](#)
- #define [ATOMIC\\_INT\\_LOCK\\_FREE](#)
- #define [ATOMIC\\_LLONG\\_LOCK\\_FREE](#)
- #define [ATOMIC\\_LONG\\_LOCK\\_FREE](#)
- #define [ATOMIC\\_POINTER\\_LOCK\\_FREE](#)
- #define [ATOMIC\\_SHORT\\_LOCK\\_FREE](#)
- #define [ATOMIC\\_WCHAR\\_T\\_LOCK\\_FREE](#)

## 5.11.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<atomic>`.

## 5.12 atomic\_word.h File Reference

## Typedefs

- typedef int **\_Atomic\_word**

## 5.12.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.13 atomicity.h File Reference

## Namespaces

- [\\_\\_gnu\\_cxx](#)

## Macros

- `#define _GLIBCXX_READ_MEM_BARRIER`
- `#define _GLIBCXX_WRITE_MEM_BARRIER`

## Functions

- static void **\_\_gnu\_cxx::\_\_atomic\_add\_single** (\_Atomic\_word \* \_\_mem, int \_\_val)
- else **\_\_gnu\_cxx::\_\_atomic\_add\_single** (\_\_mem, \_\_val)
- \_Atomic\_word **\_\_gnu\_cxx::\_\_attribute\_\_** ((\_\_unused\_\_)) \_\_exchange\_and\_add(volatile \_Atomic\_word \*
- static \_Atomic\_word **\_\_gnu\_cxx::\_\_exchange\_and\_add\_single** (\_Atomic\_word \* \_\_mem, int \_\_val)
- else return **\_\_gnu\_cxx::\_\_exchange\_and\_add\_single** (\_\_mem, \_\_val)
- static \_Atomic\_word int \_\_val **\_\_gnu\_cxx::if** (\_\_gthread\_active\_p()) return \_\_exchange\_and\_add(\_\_mem
- \_Atomic\_word int **\_\_gnu\_cxx::throw** ()

## Variables

- static \_Atomic\_word int \_\_val **\_\_gnu\_cxx::\_\_val**

## 5.13.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.



## 5.14 auto\_ptr.h File Reference

### Classes

- class [std::auto\\_ptr<\\_Tp>](#)
- struct [std::auto\\_ptr\\_ref<\\_Tp1>](#)

### Namespaces

- [std](#)

### 5.14.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 5.15 backward\_warning.h File Reference

### 5.15.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

## 5.16 balanced\_quicksort.h File Reference

### Classes

- struct [\\_\\_gnu\\_parallel::\\_\\_QSBThreadLocal<\\_RAIter>](#)

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Functions

- template<typename \_RAIter, typename \_Compare> void [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\\_qsb](#) (\_RAIter \_\_begin, \_RAIter \_\_end, \_Compare \_\_comp, \_ThreadIndex \_\_num\_threads)
- template<typename \_RAIter, typename \_Compare> void [\\_\\_gnu\\_parallel::\\_\\_qsb\\_conquer](#) (\_QSBThreadLocal<\_RAIter> \*\_\_tls, \_RAIter \_\_begin, \_RAIter \_\_end, \_Compare \_\_comp, \_ThreadIndex \_\_iam, \_ThreadIndex \_\_num\_threads, bool \_\_parent\_wait)
- template<typename \_RAIter, typename \_Compare> std::iterator\_traits<\_RAIter>::difference\_type [\\_\\_gnu\\_parallel::\\_\\_qsb\\_divide](#) (\_RAIter \_\_begin, \_RAIter \_\_end, \_Compare \_\_comp, \_ThreadIndex \_\_num\_threads)
- template<typename \_RAIter, typename \_Compare> void [\\_\\_gnu\\_parallel::\\_\\_qsb\\_local\\_sort\\_with\\_helping](#) (\_QSBThreadLocal<\_RAIter> \*\_\_tls, \_Compare &\_\_comp, \_ThreadIndex \_\_iam, bool \_\_wait)

## 5.16.1 Detailed Description

Implementation of a dynamically load-balanced parallel quicksort.

It works in-place and needs only logarithmic extra memory. The algorithm is similar to the one proposed in

P. Tsigas and Y. Zhang. A simple, fast parallel implementation of quicksort and its performance evaluation on SUN enterprise 10000. In 11th Euromicro Conference on Parallel, Distributed and Network-Based Processing, page 372, 2003.

This file is a GNU parallel extension to the Standard C++ Library.

5.17 **base.h** File Reference

## Namespaces

- [\\_\\_gnu\\_profile](#)
- [std](#)
- [std::\\_\\_profile](#)

## 5.17.1 Detailed Description

Sequential helper functions. This file is a GNU profile extension to the Standard C++ Library.

5.18 **base.h** File Reference

## Classes

- class [\\_\\_gnu\\_parallel::\\_\\_binder1st<\\_Operation, \\_FirstArgumentType, \\_SecondArgumentType, \\_ResultType >](#)
- class [\\_\\_gnu\\_parallel::\\_\\_binder2nd<\\_Operation, \\_FirstArgumentType, \\_SecondArgumentType, \\_ResultType >](#)
- class [\\_\\_gnu\\_parallel::\\_\\_unary\\_negate<\\_Predicate, argument\\_type >](#)
- class [\\_\\_gnu\\_parallel::\\_\\_EqualFromLess<\\_T1, \\_T2, \\_Compare >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_EqualTo<\\_T1, \\_T2 >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_Less<\\_T1, \\_T2 >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_Multiplies<\\_Tp1, \\_Tp2, \\_Result >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_Plus<\\_Tp1, \\_Tp2, \\_Result >](#)
- class [\\_\\_gnu\\_parallel::\\_\\_PseudoSequence<\\_Tp, \\_DifferenceTp >](#)
- class [\\_\\_gnu\\_parallel::\\_\\_PseudoSequenceIterator<\\_Tp, \\_DifferenceTp >](#)

## Namespaces

- [\\_\\_gnu\\_parallel](#)
- [\\_\\_gnu\\_sequential](#)
- [std](#)
- [std::\\_\\_parallel](#)

## Macros

- [#define \\_GLIBCXX\\_PARALLEL\\_ASSERT\(\\_Condition\)](#)

## Functions

- void [\\_\\_gnu\\_parallel::\\_\\_decode2](#) (\_CASable \_\_x, int &\_\_a, int &\_\_b)
- \_CASable [\\_\\_gnu\\_parallel::\\_\\_encode2](#) (int \_\_a, int \_\_b)
- \_ThreadIndex [\\_\\_gnu\\_parallel::\\_\\_get\\_max\\_threads](#) ()
- bool [\\_\\_gnu\\_parallel::\\_\\_is\\_parallel](#) (const \_Parallelism \_\_p)
- template<typename \_RAlter, typename \_Compare > \_RAlter [\\_\\_gnu\\_parallel::\\_\\_median\\_of\\_three\\_iterators](#) (\_RAlter \_\_a, \_RAlter \_\_b, \_RAlter \_\_c, \_Compare \_\_comp)
- template<typename \_Size > \_Size [\\_\\_gnu\\_parallel::\\_\\_rd\\_log2](#) (\_Size \_\_n)
- template<typename \_Tp > const \_Tp & [\\_\\_gnu\\_parallel::max](#) (const \_Tp &\_\_a, const \_Tp &\_\_b)
- template<typename \_Tp > const \_Tp & [\\_\\_gnu\\_parallel::min](#) (const \_Tp &\_\_a, const \_Tp &\_\_b)

### 5.18.1 Detailed Description

Sequential helper functions. This file is a GNU parallel extension to the Standard C++ Library.

## 5.19 [basic\\_file.h](#) File Reference

### Namespaces

- [std](#)

### 5.19.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

## 5.20 [basic\\_ios.h](#) File Reference

### Classes

- class [std::basic\\_ios<\\_CharT, \\_Traits >](#)

### Namespaces

- [std](#)

### Functions

- template<typename \_Facet > const \_Facet & [std::\\_\\_check\\_facet](#) (const \_Facet \*\_\_f)

### 5.20.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

## 5.21 `basic_iterator.h` File Reference

### 5.21.1 Detailed Description

Includes the original header files concerned with iterators except for stream iterators. This file is a GNU parallel extension to the Standard C++ Library.

## 5.22 `basic_string.h` File Reference

### Classes

- class `std::basic_string<_CharT, _Traits, _Alloc>`
- struct `std::hash<string>`
- struct `std::hash<u16string>`
- struct `std::hash<u32string>`
- struct `std::hash<wstring>`

### Namespaces

- `std`

### Functions

- template<typename \_CharT, typename \_Traits, typename \_Alloc> `basic_istream<_CharT, _Traits>` & `std::getline` (`basic_istream<_CharT, _Traits>` & \_\_is, `basic_string<_CharT, _Traits, _Alloc>` & \_\_str, `_CharT` \_\_delim)
- template<typename \_CharT, typename \_Traits, typename \_Alloc> `basic_istream<_CharT, _Traits>` & `std::getline` (`basic_istream<_CharT, _Traits>` & \_\_is, `basic_string<_CharT, _Traits, _Alloc>` & \_\_str)
- template<> `basic_istream<char>` & `std::getline` (`basic_istream<char>` & \_\_in, `basic_string<char>` & \_\_str, `char` \_\_delim)
- template<> `basic_istream<wchar_t>` & `std::getline` (`basic_istream<wchar_t>` & \_\_in, `basic_string<wchar_t>` & \_\_str, `wchar_t` \_\_delim)
- template<typename \_CharT, typename \_Traits, typename \_Alloc> `bool` `std::operator!=` (`const basic_string<_CharT, _Traits, _Alloc>` & \_\_lhs, `const basic_string<_CharT, _Traits, _Alloc>` & \_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Alloc> `bool` `std::operator!=` (`const _CharT *` \_\_lhs, `const basic_string<_CharT, _Traits, _Alloc>` & \_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Alloc> `bool` `std::operator!=` (`const basic_string<_CharT, _Traits, _Alloc>` & \_\_lhs, `const _CharT *` \_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Alloc> `basic_string<_CharT, _Traits, _Alloc>` & `std::operator+` (`const basic_string<_CharT, _Traits, _Alloc>` & \_\_lhs, `const basic_string<_CharT, _Traits, _Alloc>` & \_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Alloc> `basic_string<_CharT, _Traits, _Alloc>` & `std::operator+` (`const _CharT *` \_\_lhs, `const basic_string<_CharT, _Traits, _Alloc>` & \_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Alloc> `basic_string<_CharT, _Traits, _Alloc>` & `std::operator+` (`_CharT` \_\_lhs, `const basic_string<_CharT, _Traits, _Alloc>` & \_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Alloc> `basic_string<_CharT, _Traits, _Alloc>` & `std::operator+` (`const basic_string<_CharT, _Traits, _Alloc>` & \_\_lhs, `const _CharT *` \_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Alloc> `basic_string<_CharT, _Traits, _Alloc>` & `std::operator+` (`const basic_string<_CharT, _Traits, _Alloc>` & \_\_lhs, `_CharT` \_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Alloc> `basic_string<_CharT, _Traits, _Alloc>` & `std::operator+` (`basic_string<_CharT, _Traits, _Alloc>` && \_\_lhs, `const basic_string<_CharT, _Traits, _Alloc>` & \_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Alloc> `basic_string<_CharT, _Traits, _Alloc>` & `std::operator+` (`const basic_string<_CharT, _Traits, _Alloc>` & \_\_lhs, `basic_string<_CharT, _Traits, _Alloc>` && \_\_rhs)



### 5.22.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

## 5.23 bin\_search\_tree\_.hpp File Reference

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_ASSERT_NODE_CONSISTENT(_Node)`
- `#define PB_DS_BIN_TREE_NAME`
- `#define PB_DS_BIN_TREE_TRAITS_BASE`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_STRUCT_ONLY_ASSERT_VALID(X)`

### 5.23.1 Detailed Description

Contains an implementation class for binary search tree.

## 5.24 binary\_heap\_.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::binary\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_ASSERT_VALID(X)`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_DEBUG_VERIFY(_Cond)`
- `#define PB_DS_ENTRY_CMP_DEC`
- `#define PB_DS_RESIZE_POLICY_DEC`

### 5.24.1 Detailed Description

Contains an implementation class for a binary heap.

## 5.25 binders.h File Reference

### Classes

- class [std::binder1st<\\_Operation>](#)
- class [std::binder2nd<\\_Operation>](#)

### Namespaces

- [std](#)

### Functions

- `template<typename _Operation, typename _Tp> binder1st<_Operation> std::bind1st (const _Operation &__fn, const _Tp &__x)`
- `template<typename _Operation, typename _Tp> binder2nd<_Operation> std::bind2nd (const _Operation &__fn, const _Tp &__x)`

#### 5.25.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 5.26 binomial\_heap\_.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::binomial\\_heap<Value\\_Type, Cmp\\_Fn, \\_Alloc>](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

#### 5.26.1 Detailed Description

Contains an implementation class for a binomial heap.

## 5.27 binomial\_heap\_base\_.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::binomial\\_heap\\_base<Value\\_Type, Cmp\\_Fn, \\_Alloc>](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- #define **PB\_DS\_ASSERT\_BASE\_NODE\_CONSISTENT**(\_Node, \_Bool)
- #define **PB\_DS\_ASSERT\_VALID\_COND**(X, \_StrictlyBinomial)
- #define **PB\_DS\_B\_HEAP\_BASE**
- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**

## 5.27.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

## 5.28 bitmap\_allocator.h File Reference

## Classes

- class [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_mini\\_vector<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_Bitmap\\_counter<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_Ffit\\_finder<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::bitmap\\_allocator<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::bitmap\\_allocator<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::free\\_list](#)

## Namespaces

- [\\_\\_gnu\\_cxx](#)
- [\\_\\_gnu\\_cxx::\\_\\_detail](#)

## Macros

- #define [\\_BALLOC\\_ALIGN\\_BYTES](#)

## Enumerations

- enum { **bits\_per\_byte**, **bits\_per\_block** }

## Functions

- void [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_bit\\_allocate](#) (size\_t \* \_\_pmap, size\_t \_\_pos) throw ()
- void [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_bit\\_free](#) (size\_t \* \_\_pmap, size\_t \_\_pos) throw ()
- template<typename \_ForwardIterator, typename \_Tp, typename \_Compare> \_ForwardIterator [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_lower\\_bound](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp & \_\_val, \_Compare \_\_comp)
- template<typename \_AddrPair> size\_t [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_num\\_bitmaps](#) (\_AddrPair \_\_ap)
- template<typename \_AddrPair> size\_t [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_num\\_blocks](#) (\_AddrPair \_\_ap)



- `size_t __gnu_cxx::Bit_scan_forward (size_t __num)`
- `template<typename _Tp1, typename _Tp2> bool __gnu_cxx::operator!= (const bitmap_allocator< _Tp1 > &, const bitmap_allocator< _Tp2 > &) throw ()`
- `template<typename _Tp1, typename _Tp2> bool __gnu_cxx::operator== (const bitmap_allocator< _Tp1 > &, const bitmap_allocator< _Tp2 > &) throw ()`

### 5.28.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

### 5.28.2 Macro Definition Documentation

#### 5.28.2.1 `#define _BALLOC_ALIGN_BYTES`

The constant in the expression below is the alignment required in bytes.

Definition at line 43 of file `bitmap_allocator.h`.

## 5.29 `boost_concept_check.h` File Reference

### Namespaces

- `__gnu_cxx`

### Macros

- `#define _GLIBCXX_CLASS_REQUIRES(_type_var, _ns, _concept)`
- `#define _GLIBCXX_CLASS_REQUIRES2(_type_var1, _type_var2, _ns, _concept)`
- `#define _GLIBCXX_CLASS_REQUIRES3(_type_var1, _type_var2, _type_var3, _ns, _concept)`
- `#define _GLIBCXX_CLASS_REQUIRES4(_type_var1, _type_var2, _type_var3, _type_var4, _ns, _concept)`
- `#define _GLIBCXX_DEFINE_BINARY_OPERATOR_CONSTRAINT(_OP, _NAME)`
- `#define _GLIBCXX_DEFINE_BINARY_PREDICATE_OP_CONSTRAINT(_OP, _NAME)`
- `#define _IsUnused`

### Functions

- `template<class _Tp> void __gnu_cxx::__aux_require_boolean_expr (const _Tp &__t)`
- `void __gnu_cxx::__error_type_must_be_a_signed_integer_type ()`
- `void __gnu_cxx::__error_type_must_be_an_integer_type ()`
- `void __gnu_cxx::__error_type_must_be_an_unsigned_integer_type ()`
- `template<class _Concept> void __gnu_cxx::__function_requires ()`

### 5.29.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

## 5.30 `branch_policy.hpp` File Reference

### Classes

- struct `__gnu_pbds::detail::branch_policy< Node_Cltr, Node_Itr, _Alloc >`
- struct `__gnu_pbds::detail::branch_policy< Node_Cltr, Node_Cltr, _Alloc >`

### Namespaces

- `__gnu_pbds`

#### 5.30.1 Detailed Description

Contains a base class for branch policies.

## 5.31 `c++0x_warning.h` File Reference

#### 5.31.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

## 5.32 `c++allocator.h` File Reference

### Namespaces

- `std`

### Typedefs

- `template<typename _Tp> using std::__allocator_base = __gnu_cxx::new_allocator< _Tp >`

#### 5.32.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 5.33 `c++config.h` File Reference

### Namespaces

- `std`

### Macros

- `#define __GLIBCXX__`
- `#define __glibcxx_assert(_Condition)`
- `#define __N(msgid)`

- #define \_GLIBCXX\_ABI\_TAG\_CXX11
- #define \_GLIBCXX\_ATOMIC\_BUILTINS
- #define \_GLIBCXX\_BEGIN\_EXTERN\_C
- #define \_GLIBCXX\_BEGIN\_NAMESPACE\_LDBL
- #define \_GLIBCXX\_BEGIN\_NAMESPACE\_VERSION
- #define \_GLIBCXX\_DEPRECATED
- #define \_GLIBCXX\_END\_EXTERN\_C
- #define \_GLIBCXX\_END\_NAMESPACE\_LDBL
- #define \_GLIBCXX\_END\_NAMESPACE\_VERSION
- #define \_GLIBCXX\_EXTERN\_TEMPLATE
- #define \_GLIBCXX\_FAST\_MATH
- #define \_GLIBCXX\_FULLY\_DYNAMIC\_STRING
- #define \_GLIBCXX\_HAVE\_\_\_CXA\_THREAD\_ATEXIT\_IMPL
- #define \_GLIBCXX\_HAVE\_ACOSF
- #define \_GLIBCXX\_HAVE\_ACOSL
- #define \_GLIBCXX\_HAVE\_AS\_SYMVER\_DIRECTIVE
- #define \_GLIBCXX\_HAVE\_ASINF
- #define \_GLIBCXX\_HAVE\_ASINL
- #define \_GLIBCXX\_HAVE\_AT\_QUICK\_EXIT
- #define \_GLIBCXX\_HAVE\_ATAN2F
- #define \_GLIBCXX\_HAVE\_ATAN2L
- #define \_GLIBCXX\_HAVE\_ATANF
- #define \_GLIBCXX\_HAVE\_ATANL
- #define \_GLIBCXX\_HAVE\_ATTRIBUTE\_VISIBILITY
- #define \_GLIBCXX\_HAVE\_CEILF
- #define \_GLIBCXX\_HAVE\_CEILL
- #define \_GLIBCXX\_HAVE\_COMPLEX\_H
- #define \_GLIBCXX\_HAVE\_COSF
- #define \_GLIBCXX\_HAVE\_COSHF
- #define \_GLIBCXX\_HAVE\_COSHL
- #define \_GLIBCXX\_HAVE\_COSL
- #define \_GLIBCXX\_HAVE\_DLFCN\_H
- #define \_GLIBCXX\_HAVE\_EBADMSG
- #define \_GLIBCXX\_HAVE\_ECANCELED
- #define \_GLIBCXX\_HAVE\_ECHILD
- #define \_GLIBCXX\_HAVE\_EIDRM
- #define \_GLIBCXX\_HAVE\_ENDIAN\_H
- #define \_GLIBCXX\_HAVE\_ENODATA
- #define \_GLIBCXX\_HAVE\_ENOLINK
- #define \_GLIBCXX\_HAVE\_ENOSPC
- #define \_GLIBCXX\_HAVE\_ENOSR
- #define \_GLIBCXX\_HAVE\_ENOSTR
- #define \_GLIBCXX\_HAVE\_ENOTRECOVERABLE
- #define \_GLIBCXX\_HAVE\_ENOTSUP
- #define \_GLIBCXX\_HAVE\_EOVERFLOW
- #define \_GLIBCXX\_HAVE\_EOWNERDEAD
- #define \_GLIBCXX\_HAVE\_EPERM
- #define \_GLIBCXX\_HAVE\_EPROTO
- #define \_GLIBCXX\_HAVE\_ETIME
- #define \_GLIBCXX\_HAVE\_ETIMEDOUT
- #define \_GLIBCXX\_HAVE\_ETXTBSY

- #define \_GLIBCXX\_HAVE\_EWOULDBLOCK
- #define \_GLIBCXX\_HAVE\_EXECINFO\_H
- #define \_GLIBCXX\_HAVE\_EXPF
- #define \_GLIBCXX\_HAVE\_EXPL
- #define \_GLIBCXX\_HAVE\_FABSF
- #define \_GLIBCXX\_HAVE\_FABSL
- #define \_GLIBCXX\_HAVE\_FENV\_H
- #define \_GLIBCXX\_HAVE\_FINITE
- #define \_GLIBCXX\_HAVE\_FINITEF
- #define \_GLIBCXX\_HAVE\_FINITEL
- #define \_GLIBCXX\_HAVE\_FLOAT\_H
- #define \_GLIBCXX\_HAVE\_FLOORF
- #define \_GLIBCXX\_HAVE\_FLOORL
- #define \_GLIBCXX\_HAVE\_FMODF
- #define \_GLIBCXX\_HAVE\_FMODL
- #define \_GLIBCXX\_HAVE\_FREXPF
- #define \_GLIBCXX\_HAVE\_FREXPL
- #define \_GLIBCXX\_HAVE\_GETIPINFO
- #define \_GLIBCXX\_HAVE\_GETS
- #define \_GLIBCXX\_HAVE\_HYPOT
- #define \_GLIBCXX\_HAVE\_HYPOTF
- #define \_GLIBCXX\_HAVE\_HYPOTL
- #define \_GLIBCXX\_HAVE\_ICONv
- #define \_GLIBCXX\_HAVE\_INT64\_T
- #define \_GLIBCXX\_HAVE\_INT64\_T\_LONG
- #define \_GLIBCXX\_HAVE\_INTPTR\_T
- #define \_GLIBCXX\_HAVE\_ISINF
- #define \_GLIBCXX\_HAVE\_ISINFF
- #define \_GLIBCXX\_HAVE\_ISINFL
- #define \_GLIBCXX\_HAVE\_ISNAN
- #define \_GLIBCXX\_HAVE\_ISNANF
- #define \_GLIBCXX\_HAVE\_ISNANL
- #define \_GLIBCXX\_HAVE\_ISWBLANK
- #define \_GLIBCXX\_HAVE\_LC\_MESSAGES
- #define \_GLIBCXX\_HAVE\_LDEXPF
- #define \_GLIBCXX\_HAVE\_LDEXPL
- #define \_GLIBCXX\_HAVE\_LIBINTL\_H
- #define \_GLIBCXX\_HAVE\_LIMIT\_AS
- #define \_GLIBCXX\_HAVE\_LIMIT\_DATA
- #define \_GLIBCXX\_HAVE\_LIMIT\_FSIZE
- #define \_GLIBCXX\_HAVE\_LIMIT\_RSS
- #define \_GLIBCXX\_HAVE\_LIMIT\_VMEM
- #define \_GLIBCXX\_HAVE\_LINUX\_FUTEX
- #define \_GLIBCXX\_HAVE\_LOCALE\_H
- #define \_GLIBCXX\_HAVE\_LOG10F
- #define \_GLIBCXX\_HAVE\_LOG10L
- #define \_GLIBCXX\_HAVE\_LOGF
- #define \_GLIBCXX\_HAVE\_LOGL
- #define \_GLIBCXX\_HAVE\_MBSTATE\_T
- #define \_GLIBCXX\_HAVE\_MEMORY\_H
- #define \_GLIBCXX\_HAVE\_MODF

- #define \_GLIBCXX\_HAVE\_MODFF
- #define \_GLIBCXX\_HAVE\_MODFL
- #define \_GLIBCXX\_HAVE\_POLL
- #define \_GLIBCXX\_HAVE\_POWF
- #define \_GLIBCXX\_HAVE\_POWL
- #define \_GLIBCXX\_HAVE\_QUICK\_EXIT
- #define \_GLIBCXX\_HAVE\_S\_ISREG
- #define \_GLIBCXX\_HAVE\_SETENV
- #define \_GLIBCXX\_HAVE\_SINCOS
- #define \_GLIBCXX\_HAVE\_SINCOSF
- #define \_GLIBCXX\_HAVE\_SINCOSL
- #define \_GLIBCXX\_HAVE\_SINF
- #define \_GLIBCXX\_HAVE\_SINHF
- #define \_GLIBCXX\_HAVE\_SINHL
- #define \_GLIBCXX\_HAVE\_SINL
- #define \_GLIBCXX\_HAVE\_SLEEP
- #define \_GLIBCXX\_HAVE\_SQRTF
- #define \_GLIBCXX\_HAVE\_SQRTL
- #define \_GLIBCXX\_HAVE\_STDALIGN\_H
- #define \_GLIBCXX\_HAVE\_STDBOOL\_H
- #define \_GLIBCXX\_HAVE\_STDINT\_H
- #define \_GLIBCXX\_HAVE\_STDLIB\_H
- #define \_GLIBCXX\_HAVE\_STRERROR\_L
- #define \_GLIBCXX\_HAVE\_STRERROR\_R
- #define \_GLIBCXX\_HAVE\_STRING\_H
- #define \_GLIBCXX\_HAVE\_STRINGS\_H
- #define \_GLIBCXX\_HAVE\_STRTOF
- #define \_GLIBCXX\_HAVE\_STRTOLD
- #define \_GLIBCXX\_HAVE\_STRXFRM\_L
- #define \_GLIBCXX\_HAVE\_SYMVER\_SYMBOL\_RENAMING\_RUNTIME\_SUPPORT
- #define \_GLIBCXX\_HAVE\_SYS\_IOCTL\_H
- #define \_GLIBCXX\_HAVE\_SYS\_IPC\_H
- #define \_GLIBCXX\_HAVE\_SYS\_PARAM\_H
- #define \_GLIBCXX\_HAVE\_SYS\_RESOURCE\_H
- #define \_GLIBCXX\_HAVE\_SYS\_SDT\_H
- #define \_GLIBCXX\_HAVE\_SYS\_SEM\_H
- #define \_GLIBCXX\_HAVE\_SYS\_STAT\_H
- #define \_GLIBCXX\_HAVE\_SYS\_SYSINFO\_H
- #define \_GLIBCXX\_HAVE\_SYS\_TIME\_H
- #define \_GLIBCXX\_HAVE\_SYS\_TYPES\_H
- #define \_GLIBCXX\_HAVE\_SYS\_UIO\_H
- #define \_GLIBCXX\_HAVE\_TANF
- #define \_GLIBCXX\_HAVE\_TANHF
- #define \_GLIBCXX\_HAVE\_TANHL
- #define \_GLIBCXX\_HAVE\_TANL
- #define \_GLIBCXX\_HAVE\_TGMATH\_H
- #define \_GLIBCXX\_HAVE\_TLS
- #define \_GLIBCXX\_HAVE\_UNISTD\_H
- #define \_GLIBCXX\_HAVE\_USLEEP
- #define \_GLIBCXX\_HAVE\_VFWSCANF
- #define \_GLIBCXX\_HAVE\_VSWSCANF

- #define \_GLIBCXX\_HAVE\_VWSCANF
- #define \_GLIBCXX\_HAVE\_WCHAR\_H
- #define \_GLIBCXX\_HAVE\_WCSTOF
- #define \_GLIBCXX\_HAVE\_WCTYPE\_H
- #define \_GLIBCXX\_HAVE\_WRITEV
- #define \_GLIBCXX\_HOSTED
- #define \_GLIBCXX\_ICONV\_CONST
- #define \_GLIBCXX\_INLINE\_VERSION
- #define \_GLIBCXX\_NAMESPACE\_LDBL
- #define \_GLIBCXX\_PACKAGE \_GLIBCXX\_VERSION
- #define \_GLIBCXX\_PACKAGE\_BUGREPORT
- #define \_GLIBCXX\_PACKAGE\_NAME
- #define \_GLIBCXX\_PACKAGE\_STRING
- #define \_GLIBCXX\_PACKAGE\_TARNAME
- #define \_GLIBCXX\_PACKAGE\_URL
- #define \_GLIBCXX\_PSEUDO\_VISIBILITY(V)
- #define \_GLIBCXX\_RES\_LIMITS
- #define \_GLIBCXX\_STDIO\_EOF
- #define \_GLIBCXX\_STDIO\_SEEK\_CUR
- #define \_GLIBCXX\_STDIO\_SEEK\_END
- #define \_GLIBCXX\_SYMVER
- #define \_GLIBCXX\_SYMVER\_GNU
- #define \_GLIBCXX\_SYNCHRONIZATION\_HAPPENS\_AFTER(A)
- #define \_GLIBCXX\_SYNCHRONIZATION\_HAPPENS\_BEFORE(A)
- #define \_GLIBCXX\_THROW\_OR\_ABORT(\_EXC)
- #define \_GLIBCXX\_USE\_C99
- #define \_GLIBCXX\_USE\_C99\_COMPLEX
- #define \_GLIBCXX\_USE\_C99\_COMPLEX\_TR1
- #define \_GLIBCXX\_USE\_C99\_CTYPE\_TR1
- #define \_GLIBCXX\_USE\_C99\_FENV\_TR1
- #define \_GLIBCXX\_USE\_C99\_INTTYPES\_TR1
- #define \_GLIBCXX\_USE\_C99\_INTTYPES\_WCHAR\_T\_TR1
- #define \_GLIBCXX\_USE\_C99\_MATH
- #define \_GLIBCXX\_USE\_C99\_MATH\_TR1
- #define \_GLIBCXX\_USE\_C99\_STDINT\_TR1
- #define \_GLIBCXX\_USE\_CLOCK\_GETTIME\_SYSCALL
- #define \_GLIBCXX\_USE\_CLOCK\_MONOTONIC
- #define \_GLIBCXX\_USE\_CLOCK\_REALTIME
- #define \_GLIBCXX\_USE\_DECIMAL\_FLOAT
- #define \_GLIBCXX\_USE\_DEPRECATED
- #define \_GLIBCXX\_USE\_FLOAT128
- #define \_GLIBCXX\_USE\_GET\_NPROCS
- #define \_GLIBCXX\_USE\_GETTIMEOFDAY
- #define \_GLIBCXX\_USE\_INT128
- #define \_GLIBCXX\_USE\_LFS
- #define \_GLIBCXX\_USE\_LONG\_LONG
- #define \_GLIBCXX\_USE\_NLS
- #define \_GLIBCXX\_USE\_RANDOM\_TR1
- #define \_GLIBCXX\_USE\_SC\_NPROCESSORS\_ONLN
- #define \_GLIBCXX\_USE\_WCHAR\_T
- #define \_GLIBCXX\_VERBOSE

- `#define _GLIBCXX_VISIBILITY(V)`
- `#define _GLIBCXX_WEAK_DEFINITION`
- `#define _GLIBCXX_X86_RDRAND`
- `#define _GTHREAD_USE_MUTEX_TIMEDLOCK`
- `#define LT_OBJDIR`
- `#define STDC_HEADERS`

#### Typedefs

- `typedef __PTRDIFF_TYPE__ std::ptrdiff_t`
- `typedef __SIZE_TYPE__ std::size_t`

#### Variables

- `decltype(nullptr) typedef std::nullptr_t`

#### 5.33.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

### 5.34 `c++io.h` File Reference

#### Namespaces

- [std](#)

#### Typedefs

- `typedef FILE std::__c_file`
- `typedef __gthread_mutex_t std::__c_lock`

#### 5.34.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

### 5.35 `c++locale.h` File Reference

#### Namespaces

- [std](#)

#### Macros

- `#define _GLIBCXX_C_LOCALE_GNU`
- `#define _GLIBCXX_NUM_CATEGORIES`

## Typedefs

- typedef `__locale_t` **std::\_\_c\_locale**

## Functions

- int **std::\_\_convert\_from\_v** (const `__c_locale` & `__cloc` `__attribute__((__unused__))`, char \* `__out`, const int `__size` `__attribute__((__unused__))`, const char \* `__fmt`,...)

## 5.35.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

5.36 `c++locale_internal.h` File Reference

## 5.36.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

5.37 `cast.h` File Reference

## Classes

- struct [`\_\_gnu\_cxx::\_Caster<\_ToType>`](#)

## Namespaces

- [`\_\_gnu\_cxx`](#)

## Functions

- template<typename `_ToType` , typename `_FromType` > `_ToType` [`\_\_gnu\_cxx::\_\_const\_pointer\_cast`](#) (const `_FromType` & `__arg`)
- template<typename `_ToType` , typename `_FromType` > `_ToType` [`\_\_gnu\_cxx::\_\_const\_pointer\_cast`](#) (`_FromType` \* `__arg`)
- template<typename `_ToType` , typename `_FromType` > `_ToType` [`\_\_gnu\_cxx::\_\_dynamic\_pointer\_cast`](#) (const `_FromType` & `__arg`)
- template<typename `_ToType` , typename `_FromType` > `_ToType` [`\_\_gnu\_cxx::\_\_dynamic\_pointer\_cast`](#) (`_FromType` \* `__arg`)
- template<typename `_ToType` , typename `_FromType` > `_ToType` [`\_\_gnu\_cxx::\_\_reinterpret\_pointer\_cast`](#) (const `_FromType` & `__arg`)
- template<typename `_ToType` , typename `_FromType` > `_ToType` [`\_\_gnu\_cxx::\_\_reinterpret\_pointer\_cast`](#) (`_FromType` \* `__arg`)
- template<typename `_ToType` , typename `_FromType` > `_ToType` [`\_\_gnu\_cxx::\_\_static\_pointer\_cast`](#) (const `_FromType` & `__arg`)
- template<typename `_ToType` , typename `_FromType` > `_ToType` [`\_\_gnu\_cxx::\_\_static\_pointer\_cast`](#) (`_FromType` \* `__arg`)



### 5.37.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/pointer.h>`.

## 5.38 `cc_hash_max_collision_check_resize_trigger_imp.hpp` File Reference

### 5.38.1 Detailed Description

Contains a resize trigger implementation.

## 5.39 `cc_ht_map.hpp` File Reference

### Classes

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

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CC_HASH_NAME`
- `#define PB_DS_CC_HASH_TRAITS_BASE`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_GEN_POS`
- `#define PB_DS_HASH_EQ_FN_C_DEC`
- `#define PB_DS_RANGED_HASH_FN_C_DEC`

### 5.39.1 Detailed Description

Contains an implementation class for `cc_ht_map_`.

## 5.40 `char_traits.h` File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::Char\\_types< \\_CharT >](#)
- struct [\\_\\_gnu\\_cxx::char\\_traits< \\_CharT >](#)
- struct [std::char\\_traits< \\_CharT >](#)
- struct [std::char\\_traits< char >](#)
- struct [std::char\\_traits< wchar\\_t >](#)

## Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

## 5.40.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

## 5.41 checkers.h File Reference

## Namespaces

- [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _Iter , typename _Compare > bool \_\_gnu\_parallel::\_\_is\_sorted (_Iter __begin, _Iter __end, _Compare __comp)`

## 5.41.1 Detailed Description

Routines for checking the correctness of algorithm results. This file is a GNU parallel extension to the Standard C++ Library.

## 5.42 cmp\_fn\_imps.hpp File Reference

## 5.42.1 Detailed Description

Contains implementations of `cc_ht_map_`'s entire container comparison related functions.

## 5.43 codecvt.h File Reference

## Classes

- class `std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >`
- class `std::codecvt< _InternT, _ExternT, _StateT >`
- class `std::codecvt< char, char, mbstate_t >`
- class `std::codecvt< wchar_t, char, mbstate_t >`
- class `std::codecvt_base`
- class `std::codecvt_byname< _InternT, _ExternT, _StateT >`

## Namespaces

- [std](#)

#### 5.43.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

### 5.44 `codecvt_specializations.h` File Reference

#### Classes

- struct [\\_\\_gnu\\_cxx::encoding\\_char\\_traits<\\_CharT>](#)
- class [\\_\\_gnu\\_cxx::encoding\\_state](#)
- class [std::codecvt<\\_InternT, \\_ExternT, encoding\\_state>](#)

#### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

#### Functions

- template<typename \_Tp> size\_t [std::iconv\\_adaptor](#) (size\_t(\*\_\_func)(iconv\_t, \_Tp, size\_t \*, char \*\*, size\_t \*), iconv\_t \_\_cd, char \*\*\_\_inbuf, size\_t \*\_\_inbytes, char \*\*\_\_outbuf, size\_t \*\_\_outbytes)

#### 5.44.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

### 5.45 `compatibility.h` File Reference

#### 5.45.1 Detailed Description

This is an internal header file, included by other library sources. You should not attempt to use it directly.

### 5.46 `compatibility.h` File Reference

#### Namespaces

- [\\_\\_gnu\\_parallel](#)

#### Functions

- template<typename \_Tp> \_Tp [\\_\\_gnu\\_parallel::\\_\\_add\\_omp](#) (volatile \_Tp \*\_\_ptr, \_Tp \_\_addend)
- template<typename \_Tp> bool [\\_\\_gnu\\_parallel::\\_\\_cas\\_omp](#) (volatile \_Tp \*\_\_ptr, \_Tp \_\_comparand, \_Tp \_\_← replacement)
- template<typename \_Tp> bool [\\_\\_gnu\\_parallel::\\_\\_compare\\_and\\_swap](#) (volatile \_Tp \*\_\_ptr, \_Tp \_\_comparand, \_Tp \_\_replacement)
- template<typename \_Tp> \_Tp [\\_\\_gnu\\_parallel::\\_\\_fetch\\_and\\_add](#) (volatile \_Tp \*\_\_ptr, \_Tp \_\_addend)
- void [\\_\\_gnu\\_parallel::\\_\\_yield](#) ()

## 5.46.1 Detailed Description

Compatibility layer, mostly concerned with atomic operations.

This file is a GNU parallel extension to the Standard C++ Library and contains implementation details for the library's internal use.

## 5.47 compiletime\_settings.h File Reference

## Macros

- `#define _GLIBCXX_ASSERTIONS`
- `#define _GLIBCXX_CALL(__n)`
- `#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1`
- `#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB`
- `#define _GLIBCXX_SCALE_DOWN_FPU`
- `#define _GLIBCXX_VERBOSE_LEVEL`

## 5.47.1 Detailed Description

Defines on options concerning debugging and performance, at compile-time. This file is a GNU parallel extension to the Standard C++ Library.

## 5.47.2 Macro Definition Documentation

5.47.2.1 `#define _GLIBCXX_ASSERTIONS`

Switch on many `_GLIBCXX_PARALLEL_ASSERTIONS` in parallel code. Should be switched on only locally.

Definition at line 61 of file `compiletime_settings.h`.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

5.47.2.2 `#define _GLIBCXX_CALL( __n )`

Macro to produce log message when entering a function.

## Parameters

<code>__n</code>	Input size.
------------------	-------------

## See also

`_GLIBCXX_VERBOSE_LEVEL`

Definition at line 44 of file `compiletime_settings.h`.

Referenced by `__gnu_parallel::__find_template()`, `__gnu_parallel::__for_each_template_random_access_workstealing()`, `__gnu_parallel::__merge_advance()`, `__gnu_parallel::__parallel_nth_element()`, `__gnu_parallel::__parallel_partial_sum()`, `__gnu_parallel::__parallel_partition()`, `__gnu_parallel::__parallel_random_shuffle_drs()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort_qs()`, `__gnu_parallel::__parallel_sort_qsb()`, `__gnu_parallel::__parallel_unique_copy()`, `__gnu_parallel::__search_template()`, `__gnu_parallel::__sequential_multiway_merge()`, `__gnu_parallel::__multiseq_partition()`, `__gnu_parallel::__multiseq_selection()`, `__gnu_parallel::__multiway_merge()`, `__gnu_parallel::__multiway_merge_3_variant()`, `__gnu_parallel::__multiway_merge_4_variant()`, `__gnu_parallel::__multiway_merge`

`_loser_tree()`, `__gnu_parallel::multiway_merge_loser_tree_sentinel()`, `__gnu_parallel::multiway_merge_loser_tree_↵`  
`unguarded()`, `__gnu_parallel::multiway_merge_sentinels()`, `__gnu_parallel::parallel_multiway_merge()`, and `__gnu_↵`  
`parallel::parallel_sort_mwms()`.

#### 5.47.2.3 `#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1`

Switch on many `_GLIBCXX_PARALLEL_ASSERTions` in parallel code. Consider the size of the L1 cache for `gnu_↵`  
`parallel::__parallel_random_shuffle()`.

Definition at line 68 of file `comptime_settings.h`.

#### 5.47.2.4 `#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB`

Switch on many `_GLIBCXX_PARALLEL_ASSERTions` in parallel code. Consider the size of the TLB for `gnu_parallel_↵`  
`::__parallel_random_shuffle()`.

Definition at line 74 of file `comptime_settings.h`.

#### 5.47.2.5 `#define _GLIBCXX_SCALE_DOWN_FPU`

Use floating-point scaling instead of modulo for mapping random numbers to a range. This can be faster on certain CPUs.

Definition at line 55 of file `comptime_settings.h`.

#### 5.47.2.6 `#define _GLIBCXX_VERBOSE_LEVEL`

Determine verbosity level of the parallel mode. Level 1 prints a message each time a parallel-mode function is entered.

Definition at line 37 of file `comptime_settings.h`.

## 5.48 `complex.h` File Reference

### Macros

- `#define _GLIBCXX_COMPLEX_H`

#### 5.48.1 Detailed Description

This is a Standard C++ Library header.

## 5.49 `concept_check.h` File Reference

### Macros

- `#define __glibcxx_class_requires(_a, _b)`
- `#define __glibcxx_class_requires2(_a, _b, _c)`
- `#define __glibcxx_class_requires3(_a, _b, _c, _d)`
- `#define __glibcxx_class_requires4(_a, _b, _c, _d, _e)`
- `#define __glibcxx_function_requires(...)`

#### 5.49.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

### 5.50 **concurrency.h** File Reference

#### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_scoped\\_lock](#)

#### Namespaces

- [\\_\\_gnu\\_cxx](#)

#### Enumerations

- enum **\_Lock\_policy** { **\_S\_single**, **\_S\_mutex**, **\_S\_atomic** }

#### Functions

- void **\_\_gnu\_cxx::\_\_throw\_concurrency\_lock\_error** ()
- void **\_\_gnu\_cxx::\_\_throw\_concurrency\_unlock\_error** ()

#### Variables

- static const \_Lock\_policy **\_\_gnu\_cxx::\_\_default\_lock\_policy**

#### 5.50.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

### 5.51 **cond\_dealtor.hpp** File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::detail::cond\\_dealtor< Entry, \\_Alloc >](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.51.1 Detailed Description

Contains a conditional deallocator.

## 5.52 cond\_key\_dtor\_entry\_dealtor.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::cond\\_dealtor< Entry, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### 5.52.1 Detailed Description

Contains a conditional key destructor, used for exception handling.

## 5.53 const\_iterator.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::binary\\_heap\\_const\\_iterator\\_< Value\\_Type, Entry, Simple, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- **#define PB\_DS\_BIN\_HEAP\_CIT\_BASE**

### 5.53.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

## 5.54 const\_iterator.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::left\\_child\\_next\\_sibling\\_heap\\_const\\_iterator\\_< Node, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- **#define PB\_DS\_BASIC\_HEAP\_CIT\_BASE**
- **#define PB\_DS\_CLASS\_C\_DEC**

#### 5.54.1 Detailed Description

Contains an iterator class returned by the table's `const find` and `insert` methods.

### 5.55 `const_iterator.hpp` File Reference

#### Classes

- class `const_iterator_`

#### 5.55.1 Detailed Description

Contains an iterator class used for `const` ranging over the elements of the table.

### 5.56 `constructor_destructor_fn_imps.hpp` File Reference

#### 5.56.1 Detailed Description

Contains implementations of `cc_ht_map_`'s constructors, destructor, and related functions.

### 5.57 `constructor_destructor_fn_imps.hpp` File Reference

#### 5.57.1 Detailed Description

Contains implementations of `gp_ht_map_`'s constructors, destructor, and related functions.

### 5.58 `constructor_destructor_fn_imps.hpp` File Reference

### 5.59 `constructor_destructor_no_store_hash_fn_imps.hpp` File Reference

#### 5.59.1 Detailed Description

Contains implementations of `cc_ht_map_`'s constructors, destructor, and related functions.

### 5.60 `constructor_destructor_no_store_hash_fn_imps.hpp` File Reference

#### 5.60.1 Detailed Description

Contains implementations of `gp_ht_map_`'s constructors, destructor, and related functions.

### 5.61 `constructor_destructor_store_hash_fn_imps.hpp` File Reference

#### 5.61.1 Detailed Description

Contains implementations of `cc_ht_map_`'s constructors, destructor, and related functions.



## 5.62 `constructor_destructor_store_hash_fn_imps.hpp` File Reference

### 5.62.1 Detailed Description

Contains implementations of `gp_ht_map_`'s constructors, destructor, and related functions.

## 5.63 `constructors_destructor_fn_imps.hpp` File Reference

### 5.63.1 Detailed Description

Contains an implementation class for `binary_heap_`.

## 5.64 `constructors_destructor_fn_imps.hpp` File Reference

### 5.64.1 Detailed Description

Contains an implementation for `binomial_heap_`.

## 5.65 `constructors_destructor_fn_imps.hpp` File Reference

### 5.65.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

## 5.66 `constructors_destructor_fn_imps.hpp` File Reference

### 5.66.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

## 5.67 `constructors_destructor_fn_imps.hpp` File Reference

### 5.67.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

## 5.68 `constructors_destructor_fn_imps.hpp` File Reference

### 5.68.1 Detailed Description

Contains an implementation class for `ov_tree_`.

## 5.69 `constructors_destructor_fn_imps.hpp` File Reference

### 5.69.1 Detailed Description

Contains an implementation class for a pairing heap.

## 5.70 constructors\_destructor\_fn\_imps.hpp File Reference

### 5.70.1 Detailed Description

Contains an implementation class for pat\_trie.

## 5.71 constructors\_destructor\_fn\_imps.hpp File Reference

### 5.71.1 Detailed Description

Contains an implementation for rb\_tree\_.

## 5.72 constructors\_destructor\_fn\_imps.hpp File Reference

### 5.72.1 Detailed Description

Contains an implementation for rc\_binomial\_heap\_.

## 5.73 constructors\_destructor\_fn\_imps.hpp File Reference

### 5.73.1 Detailed Description

Contains an implementation class for splay\_tree\_.

## 5.74 constructors\_destructor\_fn\_imps.hpp File Reference

### 5.74.1 Detailed Description

Contains an implementation for thin\_heap\_.

## 5.75 container\_base\_dispatch.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, cc\\_hash\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, gp\\_hash\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, list\\_update\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, ov\\_tree\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, pat\\_trie\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, rb\\_tree\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, splay\\_tree\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, cc\\_hash\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, gp\\_hash\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, list\\_update\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, ov\\_tree\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, pat\\_trie\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, rb\\_tree\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, splay\\_tree\\_tag, Policy\\_TI >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_ASSERT_VALID(X)`
- `#define PB_DS_CHECK_KEY_DOES_NOT_EXIST(_Key)`
- `#define PB_DS_CHECK_KEY_EXISTS(_Key)`
- `#define PB_DS_DATA_FALSE_INDICATOR`
- `#define PB_DS_DATA_TRUE_INDICATOR`
- `#define PB_DS_DEBUG_VERIFY(_Cond)`
- `#define PB_DS_EP2VP(X)`
- `#define PB_DS_EP2VP(X)`
- `#define PB_DS_V2F(X)`
- `#define PB_DS_V2F(X)`
- `#define PB_DS_V2S(X)`
- `#define PB_DS_V2S(X)`

### 5.75.1 Detailed Description

Contains associative container dispatching.

## 5.76 `cpp_type_traits.h` File Reference

### Classes

- class [std::move\\_iterator<\\_Iterator>](#)

## Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

### 5.76.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/type_traits>`.

## 5.77 `cpu_defines.h` File Reference

### 5.77.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

## 5.78 ctype\_base.h File Reference

### Classes

- struct [std::ctype\\_base](#)

### Namespaces

- [std](#)

#### 5.78.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 5.79 ctype\_inline.h File Reference

### Namespaces

- [std](#)

#### 5.79.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 5.80 cxxabi.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::recursive\\_init\\_error](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [abi](#)

### Typedefs

- typedef `__cxa_cdtor_return_type(* __cxxabiv1::__cxa_cdtor_type) (void *)`

### Functions

- `__cxa_dependent_exception * __cxxabiv1::__cxa_allocate_dependent_exception () noexcept`
- `void * __cxxabiv1::__cxa_allocate_exception (size_t) noexcept`
- `int __cxxabiv1::__cxa_atexit (void(*) (void *), void *, void *) noexcept`
- `void __cxxabiv1::__cxa_bad_cast () __attribute__((__noreturn__))`
- `void __cxxabiv1::__cxa_bad_typeid () __attribute__((__noreturn__))`

- void \* **\_\_cxxabiv1::\_\_cxa\_begin\_catch** (void \*) noexcept
- std::type\_info \* **\_\_cxxabiv1::\_\_cxa\_current\_exception\_type** () noexcept \_\_attribute\_\_((\_\_pure\_\_))
- void **\_\_cxxabiv1::\_\_cxa\_deleted\_virtual** (void) \_\_attribute\_\_((\_\_noreturn\_\_))
- char \* **\_\_cxxabiv1::\_\_cxa\_demangle** (const char \* \_\_mangled\_name, char \* \_\_output\_buffer, size\_t \* \_\_length, int \* \_\_status)
- void **\_\_cxxabiv1::\_\_cxa\_end\_catch** ()
- int **\_\_cxxabiv1::\_\_cxa\_finalize** (void \*)
- void **\_\_cxxabiv1::\_\_cxa\_free\_dependent\_exception** (\_\_cxa\_dependent\_exception \*) noexcept
- void **\_\_cxxabiv1::\_\_cxa\_free\_exception** (void \*) noexcept
- void \* **\_\_cxxabiv1::\_\_cxa\_get\_exception\_ptr** (void \*) noexcept \_\_attribute\_\_((\_\_pure\_\_))
- \_\_cxa\_eh\_globals \* **\_\_cxxabiv1::\_\_cxa\_get\_globals** () noexcept \_\_attribute\_\_((\_\_const\_\_))
- \_\_cxa\_eh\_globals \* **\_\_cxxabiv1::\_\_cxa\_get\_globals\_fast** () noexcept \_\_attribute\_\_((\_\_const\_\_))
- void **\_\_cxxabiv1::\_\_cxa\_guard\_abort** (\_\_guard \*) noexcept
- int **\_\_cxxabiv1::\_\_cxa\_guard\_acquire** (\_\_guard \*)
- void **\_\_cxxabiv1::\_\_cxa\_guard\_release** (\_\_guard \*) noexcept
- void **\_\_cxxabiv1::\_\_cxa\_pure\_virtual** (void) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **\_\_cxxabiv1::\_\_cxa\_rethrow** () \_\_attribute\_\_((\_\_noreturn\_\_))
- int **\_\_cxxabiv1::\_\_cxa\_thread\_atexit** (void (\*)(void \*), void \*, void \*) noexcept
- void **\_\_cxxabiv1::\_\_cxa\_throw** (void \*, std::type\_info \*, void (\*)(void \*)) \_\_attribute\_\_((\_\_noreturn\_\_))
- \_\_cxa\_vec\_ctor\_return\_type **\_\_cxxabiv1::\_\_cxa\_vec\_ctor** (void \* \_\_dest\_array, void \* \_\_src\_array, size\_t \_\_↵  
element\_count, size\_t \_\_element\_size, \_\_cxa\_ctor\_return\_type(\* \_\_constructor)(void \*, void \*), \_\_cxa\_ctor↵  
\_type \_\_destructor)
- void **\_\_cxxabiv1::\_\_cxa\_vec\_cleanup** (void \* \_\_array\_address, size\_t \_\_element\_count, size\_t \_\_s, \_\_cxa\_↵  
ctor\_type \_\_destructor) noexcept
- \_\_cxa\_vec\_ctor\_return\_type **\_\_cxxabiv1::\_\_cxa\_vec\_ctor** (void \* \_\_array\_address, size\_t \_\_element\_count,  
size\_t \_\_element\_size, \_\_cxa\_ctor\_type \_\_constructor, \_\_cxa\_ctor\_type \_\_destructor)
- void **\_\_cxxabiv1::\_\_cxa\_vec\_delete** (void \* \_\_array\_address, size\_t \_\_element\_size, size\_t \_\_padding\_size, ↵  
\_\_cxa\_ctor\_type \_\_destructor)
- void **\_\_cxxabiv1::\_\_cxa\_vec\_delete2** (void \* \_\_array\_address, size\_t \_\_element\_size, size\_t \_\_padding\_size,  
\_\_cxa\_ctor\_type \_\_destructor, void(\* \_\_dealloc)(void \*))
- void **\_\_cxxabiv1::\_\_cxa\_vec\_delete3** (void \* \_\_array\_address, size\_t \_\_element\_size, size\_t \_\_padding\_size,  
\_\_cxa\_ctor\_type \_\_destructor, void(\* \_\_dealloc)(void \*, size\_t))
- void **\_\_cxxabiv1::\_\_cxa\_vec\_dtor** (void \* \_\_array\_address, size\_t \_\_element\_count, size\_t \_\_element\_size, ↵  
\_\_cxa\_ctor\_type \_\_destructor)
- void \* **\_\_cxxabiv1::\_\_cxa\_vec\_new** (size\_t \_\_element\_count, size\_t \_\_element\_size, size\_t \_\_padding\_size,  
\_\_cxa\_ctor\_type \_\_constructor, \_\_cxa\_ctor\_type \_\_destructor)
- void \* **\_\_cxxabiv1::\_\_cxa\_vec\_new2** (size\_t \_\_element\_count, size\_t \_\_element\_size, size\_t \_\_padding\_size,  
\_\_cxa\_ctor\_type \_\_constructor, \_\_cxa\_ctor\_type \_\_destructor, void \*(\* \_\_alloc)(size\_t), void(\* \_\_dealloc)(void  
\*))
- void \* **\_\_cxxabiv1::\_\_cxa\_vec\_new3** (size\_t \_\_element\_count, size\_t \_\_element\_size, size\_t \_\_padding\_size,  
\_\_cxa\_ctor\_type \_\_constructor, \_\_cxa\_ctor\_type \_\_destructor, void \*(\* \_\_alloc)(size\_t), void(\* \_\_dealloc)(void  
\*, size\_t))
- void \* **\_\_cxxabiv1::\_\_dynamic\_cast** (const void \* \_\_src\_ptr, const \_\_class\_type\_info \* \_\_src\_type, const \_\_↵  
class\_type\_info \* \_\_dst\_type, ptrdiff\_t \_\_src2dst)

### 5.80.1 Detailed Description

The header provides an interface to the C++ ABI.

## 5.81 `cxxabi_forced.h` File Reference

### Classes

- class [\\_\\_cxxabiv1::\\_\\_forced\\_unwind](#)

#### 5.81.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<cxxabi.h>`.

## 5.82 `cxxabi_tweaks.h` File Reference

### Macros

- `#define _GLIBCXX_CXA_VEC_CTOR_RETURN(x)`
- `#define _GLIBCXX_GUARD_BIT`
- `#define _GLIBCXX_GUARD_PENDING_BIT`
- `#define _GLIBCXX_GUARD_SET(x)`
- `#define _GLIBCXX_GUARD_TEST(x)`
- `#define _GLIBCXX_GUARD_WAITING_BIT`

### Typedefs

- typedef void `__cxxabiv1::__cxa_cdtor_return_type`
- typedef void `__cxxabiv1::__cxa_vec_ctor_return_type`

### Functions

- `__extension__ typedef int __guard __cxxabiv1::__attribute__ ((mode(__DI__)))`

#### 5.82.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<cxxabi.h>`.

## 5.83 `debug.h` File Reference

### Namespaces

- [\\_\\_gnu\\_debug](#)
- [std](#)
- [std::\\_\\_debug](#)

## Macros

- `#define __glibcxx_requires_cond(_Cond, _Msg)`
- `#define __glibcxx_requires_heap(_First, _Last)`
- `#define __glibcxx_requires_heap_pred(_First, _Last, _Pred)`
- `#define __glibcxx_requires_non_empty_range(_First, _Last)`
- `#define __glibcxx_requires_nonempty()`
- `#define __glibcxx_requires_partitioned_lower(_First, _Last, _Value)`
- `#define __glibcxx_requires_partitioned_lower_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_requires_partitioned_upper(_First, _Last, _Value)`
- `#define __glibcxx_requires_partitioned_upper_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_requires_sorted(_First, _Last)`
- `#define __glibcxx_requires_sorted_pred(_First, _Last, _Pred)`
- `#define __glibcxx_requires_sorted_set(_First1, _Last1, _First2)`
- `#define __glibcxx_requires_sorted_set_pred(_First1, _Last1, _First2, _Pred)`
- `#define __glibcxx_requires_string(_String)`
- `#define __glibcxx_requires_string_len(_String, _Len)`
- `#define __glibcxx_requires_subscript(_N)`
- `#define __glibcxx_requires_valid_range(_First, _Last)`
- `#define __GLIBCXX_DEBUG_ASSERT(_Condition)`
- `#define __GLIBCXX_DEBUG_ONLY(_Statement)`
- `#define __GLIBCXX_DEBUG_PEDASSERT(_Condition)`

### 5.83.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.84 `debug_allocator.h` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::debug\\_allocator< \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### 5.84.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.85 `debug_fn_imps.hpp` File Reference

### 5.85.1 Detailed Description

Contains an implementation class for a binary\_heap.

## 5.86 debug\_fn\_imps.hpp File Reference

### 5.86.1 Detailed Description

Contains an implementation for binomial\_heap\_.

## 5.87 debug\_fn\_imps.hpp File Reference

### 5.87.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

## 5.88 debug\_fn\_imps.hpp File Reference

### 5.88.1 Detailed Description

Contains an implementation class for bin\_search\_tree\_.

## 5.89 debug\_fn\_imps.hpp File Reference

### 5.89.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s debug-mode functions.

## 5.90 debug\_fn\_imps.hpp File Reference

### 5.90.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s debug-mode functions.

## 5.91 debug\_fn\_imps.hpp File Reference

### 5.91.1 Detailed Description

Contains an implementation class for left\_child\_next\_sibling\_heap\_.

## 5.92 debug\_fn\_imps.hpp File Reference

### 5.92.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s debug-mode functions.

## 5.93 debug\_fn\_imps.hpp File Reference

### 5.93.1 Detailed Description

Contains an implementation class for ov\_tree\_.



### 5.94 `debug_fn_imps.hpp` File Reference

#### 5.94.1 Detailed Description

Contains an implementation class for a pairing heap.

### 5.95 `debug_fn_imps.hpp` File Reference

#### 5.95.1 Detailed Description

Contains an implementation class for `pat_trie_`.

### 5.96 `debug_fn_imps.hpp` File Reference

#### 5.96.1 Detailed Description

Contains an implementation for `rb_tree_`.

### 5.97 `debug_fn_imps.hpp` File Reference

#### 5.97.1 Detailed Description

Contains an implementation for `rc_binomial_heap_`.

### 5.98 `debug_fn_imps.hpp` File Reference

#### 5.98.1 Detailed Description

Contains an implementation class for `splay_tree_`.

### 5.99 `debug_fn_imps.hpp` File Reference

#### 5.99.1 Detailed Description

Contains an implementation for `thin_heap_`.

### 5.100 `debug_map_base.hpp` File Reference

#### 5.100.1 Detailed Description

Contains a debug-mode base for all maps.

### 5.101 `debug_no_store_hash_fn_imps.hpp` File Reference

#### 5.101.1 Detailed Description

Contains implementations of `cc_ht_map_`'s debug-mode functions.

## 5.102 debug\_no\_store\_hash\_fn\_imps.hpp File Reference

### 5.102.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s debug-mode functions.

## 5.103 debug\_store\_hash\_fn\_imps.hpp File Reference

### 5.103.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s debug-mode functions.

## 5.104 debug\_store\_hash\_fn\_imps.hpp File Reference

### 5.104.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s debug-mode functions.

## 5.105 direct\_mask\_range\_hashing\_imp.hpp File Reference

### 5.105.1 Detailed Description

Contains a range-hashing policy implementation

## 5.106 direct\_mod\_range\_hashing\_imp.hpp File Reference

### 5.106.1 Detailed Description

Contains a range-hashing policy implementation

## 5.107 enc\_filebuf.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::enc\\_filebuf<\\_CharT>](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### 5.107.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.108 entry\_cmp.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::entry\\_cmp<\\_VTp, Cmp\\_Fn, \\_Alloc, No\\_Throw >](#)
- struct [\\_\\_gnu\\_pbds::detail::entry\\_cmp<\\_VTp, Cmp\\_Fn, \\_Alloc, false >](#)
- struct [\\_\\_gnu\\_pbds::detail::entry\\_cmp<\\_VTp, Cmp\\_Fn, \\_Alloc, false >::type](#)
- struct [\\_\\_gnu\\_pbds::detail::entry\\_cmp<\\_VTp, Cmp\\_Fn, \\_Alloc, true >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.108.1 Detailed Description

Contains an implementation class for a binary\_heap.

## 5.109 entry\_list\_fn\_imps.hpp File Reference

#### 5.109.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s entry-list related functions.

## 5.110 entry\_metadata\_base.hpp File Reference

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.110.1 Detailed Description

Contains an implementation for a list update map.

## 5.111 entry\_pred.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::entry\\_pred<\\_VTp, Pred, \\_Alloc, No\\_Throw >](#)
- struct [\\_\\_gnu\\_pbds::detail::entry\\_pred<\\_VTp, Pred, \\_Alloc, false >](#)
- struct [\\_\\_gnu\\_pbds::detail::entry\\_pred<\\_VTp, Pred, \\_Alloc, true >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.111.1 Detailed Description

Contains an implementation class for a binary\_heap.

## 5.112 `eq_by_less.hpp` File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::eq\\_by\\_less](#)< Key, Cmp\_Fn >

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.112.1 Detailed Description

Contains an equivalence function.

## 5.113 `equally_split.h` File Reference

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Functions

- template<typename \_DifferenceType , typename \_OutputIterator > \_OutputIterator [\\_\\_gnu\\_parallel::\\_\\_equally\\_split](#) (\_↔ DifferenceType \_\_n, \_ThreadIndex \_\_num\_threads, \_OutputIterator \_\_s)
- template<typename \_DifferenceType > \_DifferenceType [\\_\\_gnu\\_parallel::\\_\\_equally\\_split\\_point](#) (\_DifferenceType \_\_n, ↔ \_ThreadIndex \_\_num\_threads, \_ThreadIndex \_\_thread\_no)

#### 5.113.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

## 5.114 `erase_fn_imps.hpp` File Reference

#### 5.114.1 Detailed Description

Contains an implementation class for a binary\_heap.

## 5.115 `erase_fn_imps.hpp` File Reference

#### 5.115.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

## 5.116 `erase_fn_imps.hpp` File Reference

#### 5.116.1 Detailed Description

Contains an implementation class for bin\_search\_tree\_.

### 5.117 `erase_fn_imps.hpp` File Reference

#### 5.117.1 Detailed Description

Contains implementations of `cc_ht_map_`'s erase related functions.

### 5.118 `erase_fn_imps.hpp` File Reference

#### 5.118.1 Detailed Description

Contains implementations of `gp_ht_map_`'s erase related functions.

### 5.119 `erase_fn_imps.hpp` File Reference

#### 5.119.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

### 5.120 `erase_fn_imps.hpp` File Reference

#### 5.120.1 Detailed Description

Contains implementations of `lu_map_`.

### 5.121 `erase_fn_imps.hpp` File Reference

#### 5.121.1 Detailed Description

Contains an implementation class for `ov_tree_`.

### 5.122 `erase_fn_imps.hpp` File Reference

#### 5.122.1 Detailed Description

Contains an implementation class for a pairing heap.

### 5.123 `erase_fn_imps.hpp` File Reference

#### 5.123.1 Detailed Description

Contains an implementation class for `pat_trie`.

### 5.124 `erase_fn_imps.hpp` File Reference

#### 5.124.1 Detailed Description

Contains an implementation for `rb_tree_`.

## 5.125 `erase_fn_imps.hpp` File Reference

### 5.125.1 Detailed Description

Contains an implementation for `rc_binomial_heap_`.

## 5.126 `erase_fn_imps.hpp` File Reference

### 5.126.1 Detailed Description

Contains an implementation class for `splay_tree_`.

## 5.127 `erase_fn_imps.hpp` File Reference

### 5.127.1 Detailed Description

Contains an implementation for `thin_heap_`.

## 5.128 `erase_no_store_hash_fn_imps.hpp` File Reference

### 5.128.1 Detailed Description

Contains implementations of `cc_ht_map_`'s erase related functions, when the hash value is not stored.

## 5.129 `erase_no_store_hash_fn_imps.hpp` File Reference

### 5.129.1 Detailed Description

Contains implementations of `gp_ht_map_`'s erase related functions, when the hash value is not stored.

## 5.130 `erase_store_hash_fn_imps.hpp` File Reference

### 5.130.1 Detailed Description

Contains implementations of `cc_ht_map_`'s erase related functions, when the hash value is stored.

## 5.131 `erase_store_hash_fn_imps.hpp` File Reference

### 5.131.1 Detailed Description

Contains implementations of `gp_ht_map_`'s erase related functions, when the hash value is stored.

## 5.132 `error_constants.h` File Reference

Namespaces

- [std](#)

## Enumerations

- enum `errc` { `address_family_not_supported`, `address_in_use`, `address_not_available`, `already_connected`, `argument_list_too_long`, `argument_out_of_domain`, `bad_address`, `bad_file_descriptor`, `broken_↵`  
`pipe`, `connection_aborted`, `connection_already_in_progress`, `connection_refused`, `connection_reset`,  
`cross_device_link`, `destination_address_required`, `device_or_resource_busy`, `directory_not_empty`,  
`executable_format_error`, `file_exists`, `file_too_large`, `filename_too_long`, `function_not_supported`, `host_↵`  
`unreachable`, `illegal_byte_sequence`, `inappropriate_io_control_operation`, `interrupted`, `invalid_argument`,  
`invalid_seek`, `io_error`, `is_a_directory`, `message_size`, `network_down`, `network_reset`, `network_↵`  
`unreachable`, `no_buffer_space`, `no_child_process`, `no_lock_available`, `no_message`, `no_protocol_option`,  
`no_space_on_device`, `no_such_device_or_address`, `no_such_device`, `no_such_file_or_directory`, `no_↵`  
`_such_process`, `not_a_directory`, `not_a_socket`, `not_connected`, `not_enough_memory`, `operation_in_↵`  
`progress`, `operation_not_permitted`, `operation_not_supported`, `operation_would_block`, `permission_↵`  
`denied`, `protocol_not_supported`, `read_only_file_system`, `resource_deadlock_would_occur`, `resource_↵`  
`unavailable_try_again`, `result_out_of_range`, `timed_out`, `too_many_files_open_in_system`, `too_many_↵`  
`files_open`, `too_many_links`, `too_many_symbolic_link_levels`, `wrong_protocol_type` }

## 5.132.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<system_error>`.

5.133 `exception.hpp` File Reference

## Classes

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

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Functions

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

## 5.133.1 Detailed Description

Contains exception classes.

## 5.134 `exception_defines.h` File Reference

### Macros

- `#define __catch(X)`
- `#define __throw_exception_again`
- `#define __try`

#### 5.134.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

## 5.135 `exception_ptr.h` File Reference

### Classes

- class `std::__exception_ptr::exception_ptr`

### Namespaces

- `std`

### Functions

- `template<typename _Ex> exception_ptr std::copy_exception (_Ex __ex) noexcept`
- `exception_ptr std::current_exception () noexcept`
- `template<typename _Ex> exception_ptr std::make_exception_ptr (_Ex __ex) noexcept`
- `bool std::__exception_ptr::operator!= (const exception_ptr &, const exception_ptr &) noexcept __attribute__((__pure__))`
- `bool std::__exception_ptr::operator== (const exception_ptr &, const exception_ptr &) noexcept __attribute__((__pure__))`
- `void std::rethrow_exception (exception_ptr) __attribute__((__noreturn__))`
- `void std::__exception_ptr::swap (exception_ptr &__lhs, exception_ptr &__rhs)`

#### 5.135.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

## 5.136 `extc++.h` File Reference

#### 5.136.1 Detailed Description

This is an implementation file for a precompiled header.



## 5.137 extptr\_allocator.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_ExtPtr\\_allocator< \\_Tp >](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### Functions

- template<typename \_Tp > void [\\_\\_gnu\\_cxx::swap](#) ([\\_ExtPtr\\_allocator< \\_Tp >](#) &\_\_larg, [\\_ExtPtr\\_allocator< \\_Tp >](#) &\_\_rarg)

#### 5.137.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

#### Author

Bob Walters

An example allocator which uses an alternative pointer type from bits/pointer.h. Supports test cases which confirm container support for alternative pointers.

## 5.138 features.h File Reference

### Macros

- [#define \\_GLIBCXX\\_BAL\\_QUICKSORT](#)
- [#define \\_GLIBCXX\\_FIND\\_CONSTANT\\_SIZE\\_BLOCKS](#)
- [#define \\_GLIBCXX\\_FIND\\_EQUAL\\_SPLIT](#)
- [#define \\_GLIBCXX\\_FIND\\_GROWING\\_BLOCKS](#)
- [#define \\_GLIBCXX\\_MERGESORT](#)
- [#define \\_GLIBCXX\\_QUICKSORT](#)
- [#define \\_GLIBCXX\\_TREE\\_DYNAMIC\\_BALANCING](#)
- [#define \\_GLIBCXX\\_TREE\\_FULL\\_COPY](#)
- [#define \\_GLIBCXX\\_TREE\\_INITIAL\\_SPLITTING](#)

#### 5.138.1 Detailed Description

Defines on whether to include algorithm variants.

Less variants reduce executable size and compile time. This file is a GNU parallel extension to the Standard C++ Library.

### 5.138.2 Macro Definition Documentation

#### 5.138.2.1 `#define _GLIBCXX_BAL_QUICKSORT`

Include parallel dynamically load-balanced quicksort.

See also

`__gnu_parallel::_Settings::sort_algorithm`

Definition at line 55 of file features.h.

#### 5.138.2.2 `#define _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`

Include the equal-sized blocks variant for `std::find`.

See also

`__gnu_parallel::_Settings::find_algorithm`

Definition at line 67 of file features.h.

#### 5.138.2.3 `#define _GLIBCXX_FIND_EQUAL_SPLIT`

Include the equal splitting variant for `std::find`.

See also

`__gnu_parallel::_Settings::find_algorithm`

Definition at line 74 of file features.h.

#### 5.138.2.4 `#define _GLIBCXX_FIND_GROWING_BLOCKS`

Include the growing blocks variant for `std::find`.

See also

`__gnu_parallel::_Settings::find_algorithm`

Definition at line 61 of file features.h.

#### 5.138.2.5 `#define _GLIBCXX_MERGESORT`

Include parallel multi-way mergesort.

See also

`__gnu_parallel::_Settings::sort_algorithm`

Definition at line 41 of file features.h.

#### 5.138.2.6 `#define _GLIBCXX_QUICKSORT`

Include parallel unbalanced quicksort.

See also

`__gnu_parallel::_Settings::sort_algorithm`

Definition at line 48 of file features.h.

**5.138.2.7 #define \_GLIBCXX\_TREE\_DYNAMIC\_BALANCING**

Include the dynamic balancing variant for `_Rb_tree::insert_unique(_Iter beg, _Iter __end)`.

See also

`__gnu_parallel::_Rb_tree`

Definition at line 91 of file `features.h`.

**5.138.2.8 #define \_GLIBCXX\_TREE\_FULL\_COPY**

In order to sort the input sequence of `_Rb_tree::insert_unique(_Iter beg, _Iter __end)` a full copy of the input elements is done.

See also

`__gnu_parallel::_Rb_tree`

Definition at line 100 of file `features.h`.

**5.138.2.9 #define \_GLIBCXX\_TREE\_INITIAL\_SPLITTING**

Include the initial splitting variant for `_Rb_tree::insert_unique(_Iter beg, _Iter __end)`.

See also

`__gnu_parallel::_Rb_tree`

Definition at line 83 of file `features.h`.

**5.139 fenv.h File Reference****5.139.1 Detailed Description**

This is a Standard C++ Library header.

**5.140 find.h File Reference****Namespaces**

- [\\_\\_gnu\\_parallel](#)

**Functions**

- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector> std::pair<_RAIter1, _RAIter2> \_\_gnu\_parallel::\_\_find\_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, \_Selector __selector)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector> std::pair<_RAIter1, _RAIter2> \_\_gnu\_parallel::\_\_find\_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, \_Selector __selector, equal_split_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector> std::pair<_RAIter1, _RAIter2> \_\_gnu\_parallel::\_\_find\_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, \_Selector __selector, growing_blocks_tag)`

- `template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector > std::pair< _RAIter1, _RAIter2 > \_\_gnu\_parallel::\_\_find\_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, constant_size_blocks_tag)`

#### 5.140.1 Detailed Description

Parallel implementation base for `std::find()`, `std::equal()` and related functions. This file is a GNU parallel extension to the Standard C++ Library.

### 5.141 find\_fn\_imps.hpp File Reference

#### 5.141.1 Detailed Description

Contains an implementation class for a binary\_heap.

### 5.142 find\_fn\_imps.hpp File Reference

#### 5.142.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

### 5.143 find\_fn\_imps.hpp File Reference

#### 5.143.1 Detailed Description

Contains an implementation class for bin\_search\_tree\_.

### 5.144 find\_fn\_imps.hpp File Reference

#### 5.144.1 Detailed Description

Contains implementations of `cc_ht_map_'s` find related functions.

### 5.145 find\_fn\_imps.hpp File Reference

#### 5.145.1 Detailed Description

Contains implementations of `gp_ht_map_'s` find related functions.

### 5.146 find\_fn\_imps.hpp File Reference

#### 5.146.1 Detailed Description

Contains implementations of `lu_map_`.

### 5.147 `find_fn_imps.hpp` File Reference

#### 5.147.1 Detailed Description

Contains an implementation class for a pairing heap.

### 5.148 `find_fn_imps.hpp` File Reference

#### 5.148.1 Detailed Description

Contains an implementation class for `pat_trie`.

### 5.149 `find_fn_imps.hpp` File Reference

#### 5.149.1 Detailed Description

Contains an implementation for `rb_tree_`.

### 5.150 `find_fn_imps.hpp` File Reference

#### 5.150.1 Detailed Description

Contains an implementation class for `splay_tree_`.

### 5.151 `find_fn_imps.hpp` File Reference

#### 5.151.1 Detailed Description

Contains an implementation for `thin_heap_`.

### 5.152 `find_no_store_hash_fn_imps.hpp` File Reference

#### 5.152.1 Detailed Description

Contains implementations of `gp_ht_map_`'s find related functions, when the hash value is not stored.

### 5.153 `find_selectors.h` File Reference

#### Classes

- struct [\\_\\_gnu\\_parallel::\\_\\_adjacent\\_find\\_selector](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_find\\_first\\_of\\_selector<\\_FIterator>](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_find\\_if\\_selector](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_generic\\_find\\_selector](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_mismatch\\_selector](#)

## Namespaces

- [\\_\\_gnu\\_parallel](#)

## 5.153.1 Detailed Description

\_Function objects representing different tasks to be plugged into the parallel find algorithm. This file is a GNU parallel extension to the Standard C++ Library.

## 5.154 find\_store\_hash\_fn\_imps.hpp File Reference

## 5.154.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s find related functions, when the hash value is stored.

## 5.155 find\_store\_hash\_fn\_imps.hpp File Reference

## 5.155.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s insert related functions, when the hash value is stored.

## 5.156 for\_each.h File Reference

## Namespaces

- [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _Iter , typename _UserOp , typename _Functionality , typename _Red , typename _Result > _UserOp \_\_gnu\_parallel::\_\_for\_each\_template\_random\_access (_Iter __begin, _Iter __end, _UserOp __user_op, _Functionality &__functionality, _Red __reduction, _Result __reduction_start, _Result &__output, typename std::iterator_traits<_Iter >::difference_type __bound, _Parallelism __parallelism_tag)`

## 5.156.1 Detailed Description

Main interface for embarrassingly parallel functions.

The explicit implementation are in other header files, like workstealing.h, par\_loop.h, omp\_loop.h, and omp\_loop\_↵static.h. This file is a GNU parallel extension to the Standard C++ Library.

## 5.157 for\_each\_selectors.h File Reference

## Classes

- `struct \_\_gnu\_parallel::\_\_accumulate\_binop\_reduct< _BinOp >`
- `struct \_\_gnu\_parallel::\_\_accumulate\_selector< _It >`
- `struct \_\_gnu\_parallel::\_\_adjacent\_difference\_selector< _It >`
- `struct \_\_gnu\_parallel::\_\_count\_if\_selector< _It, _Diff >`

- struct [\\_\\_gnu\\_parallel::\\_\\_count\\_selector<\\_It, \\_Diff>](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_fill\\_selector<\\_It>](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_for\\_each\\_selector<\\_It>](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_generate\\_selector<\\_It>](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_generic\\_for\\_each\\_selector<\\_It>](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_identity\\_selector<\\_It>](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_inner\\_product\\_selector<\\_It, \\_It2, \\_Tp>](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_max\\_element\\_reduct<\\_Compare, \\_It>](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_min\\_element\\_reduct<\\_Compare, \\_It>](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_replace\\_if\\_selector<\\_It, \\_Op, \\_Tp>](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_replace\\_selector<\\_It, \\_Tp>](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_transform1\\_selector<\\_It>](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_transform2\\_selector<\\_It>](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_DummyReduct](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_Nothing](#)

## Namespaces

- [\\_\\_gnu\\_parallel](#)

### 5.157.1 Detailed Description

Functors representing different tasks to be plugged into the generic parallelization methods for embarrassingly parallel functions. This file is a GNU parallel extension to the Standard C++ Library.

## 5.158 formatter.h File Reference

### Classes

- class [\\_\\_gnu\\_debug::\\_\\_Safe\\_iterator<\\_Iterator, \\_Sequence>](#)
- class [\\_\\_gnu\\_debug::\\_\\_Safe\\_local\\_iterator<\\_Iterator, \\_Sequence>](#)
- class [\\_\\_gnu\\_debug::\\_\\_Safe\\_sequence<\\_Sequence>](#)

### Namespaces

- [\\_\\_gnu\\_debug](#)

### Enumerations

- enum [\\_Debug\\_msg\\_id](#) { [\\_\\_msg\\_valid\\_range](#), [\\_\\_msg\\_insert\\_singular](#), [\\_\\_msg\\_insert\\_different](#), [\\_\\_msg\\_erase\\_bad](#), [\\_\\_msg\\_erase\\_different](#), [\\_\\_msg\\_subscript\\_oob](#), [\\_\\_msg\\_empty](#), [\\_\\_msg\\_unpartitioned](#), [\\_\\_msg\\_unpartitioned\\_pred](#), [\\_\\_msg\\_unsorted](#), [\\_\\_msg\\_unsorted\\_pred](#), [\\_\\_msg\\_not\\_heap](#), [\\_\\_msg\\_not\\_heap\\_pred](#), [\\_\\_msg\\_bad\\_bitset\\_write](#), [\\_\\_msg\\_bad\\_bitset\\_read](#), [\\_\\_msg\\_bad\\_bitset\\_flip](#), [\\_\\_msg\\_self\\_splice](#), [\\_\\_msg\\_splice\\_alloc](#), [\\_\\_msg\\_splice\\_bad](#), [\\_\\_msg\\_splice\\_other](#), [\\_\\_msg\\_splice\\_overlap](#), [\\_\\_msg\\_init\\_singular](#), [\\_\\_msg\\_init\\_copy\\_singular](#), [\\_\\_msg\\_init\\_const\\_singular](#), [\\_\\_msg\\_copy\\_singular](#), [\\_\\_msg\\_bad\\_deref](#), [\\_\\_msg\\_bad\\_inc](#), [\\_\\_msg\\_bad\\_dec](#), [\\_\\_msg\\_iter\\_subscript\\_oob](#), [\\_\\_msg\\_advance\\_oob](#), [\\_\\_msg\\_retreat\\_oob](#), [\\_\\_msg\\_iter\\_compare\\_bad](#), [\\_\\_msg\\_compare\\_different](#), [\\_\\_msg\\_iter\\_order\\_bad](#), [\\_\\_msg\\_order\\_different](#), [\\_\\_msg\\_distance\\_bad](#), [\\_\\_msg\\_distance\\_different](#), [\\_\\_msg\\_deref\\_istream](#), [\\_\\_msg\\_inc\\_istream](#), [\\_\\_msg\\_output\\_ostream](#), [\\_\\_msg\\_deref\\_istreambuf](#), [\\_\\_msg\\_inc\\_istreambuf](#), [\\_\\_msg\\_insert](#)

```
after_end, __msg_erase_after_bad, __msg_valid_range2, __msg_local_iter_compare_bad, __msg_non_↵
_empty_range, __msg_self_move_assign, __msg_bucket_index_oob, __msg_valid_load_factor, __msg_↵
_equal_allocs }
```

## Functions

- `template<typename _Iterator> bool __gnu_debug::__check_singular (_Iterator &)`

### 5.158.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.159 forward\_list.h File Reference

### Classes

- struct `std::_Fwd_list_base<_Tp, _Alloc>`
- struct `std::_Fwd_list_const_iterator<_Tp>`
- struct `std::_Fwd_list_iterator<_Tp>`
- struct `std::_Fwd_list_node<_Tp>`
- struct `std::_Fwd_list_node_base`
- class `std::forward_list<_Tp, _Alloc>`

### Namespaces

- `std`

### Functions

- `template<typename _Tp> bool std::operator!= (const _Fwd_list_iterator<_Tp> &__x, const _Fwd_list_const_↵  
iterator<_Tp> &__y)`
- `template<typename _Tp, typename _Alloc> bool std::operator!= (const forward_list<_Tp, _Alloc> &__lx, const  
forward_list<_Tp, _Alloc> &__ly)`
- `template<typename _Tp, typename _Alloc> bool std::operator< (const forward_list<_Tp, _Alloc> &__lx, const  
forward_list<_Tp, _Alloc> &__ly)`
- `template<typename _Tp, typename _Alloc> bool std::operator<= (const forward_list<_Tp, _Alloc> &__lx, const  
forward_list<_Tp, _Alloc> &__ly)`
- `template<typename _Tp> bool std::operator== (const _Fwd_list_iterator<_Tp> &__x, const _Fwd_list_const_↵  
iterator<_Tp> &__y)`
- `template<typename _Tp, typename _Alloc> bool std::operator== (const forward_list<_Tp, _Alloc> &__lx, const  
forward_list<_Tp, _Alloc> &__ly)`
- `template<typename _Tp, typename _Alloc> bool std::operator> (const forward_list<_Tp, _Alloc> &__lx, const  
forward_list<_Tp, _Alloc> &__ly)`
- `template<typename _Tp, typename _Alloc> bool std::operator>= (const forward_list<_Tp, _Alloc> &__lx, const  
forward_list<_Tp, _Alloc> &__ly)`
- `template<typename _Tp, typename _Alloc> void std::swap (forward_list<_Tp, _Alloc> &__lx, forward_list<_Tp, _Alloc  
> &__ly)`



### 5.159.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<forward_list>`.

## 5.160 `functexcept.h` File Reference

### Namespaces

- [std](#)

### Functions

- void **std::\_\_throw\_bad\_alloc** (void) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_bad\_cast** (void) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_bad\_exception** (void) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_bad\_function\_call** () `__attribute__((__noreturn__))`
- void **std::\_\_throw\_bad\_typeid** (void) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_domain\_error** (const char \*) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_future\_error** (int) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_invalid\_argument** (const char \*) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_ios\_failure** (const char \*) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_length\_error** (const char \*) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_logic\_error** (const char \*) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_out\_of\_range** (const char \*) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_overflow\_error** (const char \*) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_range\_error** (const char \*) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_runtime\_error** (const char \*) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_system\_error** (int) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_underflow\_error** (const char \*) `__attribute__((__noreturn__))`

### 5.160.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

This header provides support for `-fno-exceptions`.

## 5.161 `functional_hash.h` File Reference

### Classes

- struct [std::hash< \\_Tp >](#)
- struct [std::hash< \\_Tp \\* >](#)
- struct [std::hash< bool >](#)
- struct [std::hash< char >](#)
- struct [std::hash< char16\\_t >](#)
- struct [std::hash< char32\\_t >](#)
- struct [std::hash< double >](#)
- struct [std::hash< float >](#)

- struct [std::hash< int >](#)
- struct [std::hash< long >](#)
- struct [std::hash< long double >](#)
- struct [std::hash< long long >](#)
- struct [std::hash< short >](#)
- struct [std::hash< signed char >](#)
- struct [std::hash< unsigned char >](#)
- struct [std::hash< unsigned int >](#)
- struct [std::hash< unsigned long >](#)
- struct [std::hash< unsigned long long >](#)
- struct [std::hash< unsigned short >](#)
- struct [std::hash< wchar\\_t >](#)

## Namespaces

- [std](#)

## Macros

- `#define \_Cxx\_hashtable\_define\_trivial\_hash(_Tp)`

### 5.161.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 5.162 functions.h File Reference

### Classes

- class [\\_\\_gnu\\_debug::\\_Safe\\_iterator< \\_Iterator, \\_Sequence >](#)

### Namespaces

- [\\_\\_gnu\\_debug](#)

### Functions

- `template<typename _Iterator > \_Siter\_base< \_Iterator >::iterator\_type \_\_gnu\_debug::\_base (_Iterator __it)`
- `template<typename _Iterator > bool \_\_gnu\_debug::\_check\_dereferenceable (_Iterator &)`
- `template<typename _Tp > bool \_\_gnu\_debug::\_check\_dereferenceable (const _Tp *__ptr)`
- `template<typename _Iterator, typename _Sequence > bool \_\_gnu\_debug::\_check\_dereferenceable (const \_Safe\_iterator< \_Iterator, \_Sequence > &__x)`
- `template<typename _ForwardIterator, typename _Tp > bool \_\_gnu\_debug::\_check\_partitioned\_lower (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred > bool \_\_gnu\_debug::\_check\_partitioned\_lower (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred)`
- `template<typename _ForwardIterator, typename _Tp > bool \_\_gnu\_debug::\_check\_partitioned\_upper (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`

- `template<typename _ForwardIterator, typename _Tp, typename _Pred> bool \_\_gnu\_debug::\_\_check\_partitioned\_upper (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred)`
- `template<typename _Iterator> bool \_\_gnu\_debug::\_\_check\_singular (_Iterator &)`
- `template<typename _Tp> bool \_\_gnu\_debug::\_\_check\_singular (const _Tp * __ptr)`
- `template<typename _Iterator, typename _Sequence> bool \_\_gnu\_debug::\_\_check\_singular (const _Safe_iterator< _Iterator, _Sequence> & __x)`
- `bool \_\_gnu\_debug::\_\_check\_singular\_aux (const void *)`
- `template<typename _InputIterator> bool \_\_gnu\_debug::\_\_check\_sorted (const _InputIterator & __first, const _InputIterator & __last)`
- `template<typename _InputIterator, typename _Predicate> bool \_\_gnu\_debug::\_\_check\_sorted (const _InputIterator & __first, const _InputIterator & __last, _Predicate __pred)`
- `template<typename _InputIterator> bool \_\_gnu\_debug::\_\_check\_sorted\_aux (const _InputIterator &, const _InputIterator &, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator> bool \_\_gnu\_debug::\_\_check\_sorted\_aux (_ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `template<typename _Iterator, typename _Sequence> bool \_\_gnu\_debug::\_\_check\_sorted\_aux (const _Safe_iterator< _Iterator, _Sequence> & __first, const _Safe_iterator< _Iterator, _Sequence> & __last, std::random\_access\_iterator\_tag __tag)`
- `template<typename _InputIterator, typename _Predicate> bool \_\_gnu\_debug::\_\_check\_sorted\_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator, typename _Predicate> bool \_\_gnu\_debug::\_\_check\_sorted\_aux (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, std::forward\_iterator\_tag)`
- `template<typename _Iterator, typename _Sequence, typename _Predicate> bool \_\_gnu\_debug::\_\_check\_sorted\_aux (const _Safe_iterator< _Iterator, _Sequence> & __first, const _Safe_iterator< _Iterator, _Sequence> & __last, _Predicate __pred, std::random\_access\_iterator\_tag __tag)`
- `template<typename _InputIterator1, typename _InputIterator2> bool \_\_gnu\_debug::\_\_check\_sorted\_set (const _InputIterator1 & __first, const _InputIterator1 & __last, const _InputIterator2 &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Predicate> bool \_\_gnu\_debug::\_\_check\_sorted\_set (const _InputIterator1 & __first, const _InputIterator1 & __last, const _InputIterator2 &, _Predicate __pred)`
- `template<typename _InputIterator> bool \_\_gnu\_debug::\_\_check\_sorted\_set\_aux (const _InputIterator & __first, const _InputIterator & __last, std::true\_type)`
- `template<typename _InputIterator> bool \_\_gnu\_debug::\_\_check\_sorted\_set\_aux (const _InputIterator &, const _InputIterator &, std::false\_type)`
- `template<typename _InputIterator, typename _Predicate> bool \_\_gnu\_debug::\_\_check\_sorted\_set\_aux (const _InputIterator & __first, const _InputIterator & __last, _Predicate __pred, std::true\_type)`
- `template<typename _InputIterator, typename _Predicate> bool \_\_gnu\_debug::\_\_check\_sorted\_set\_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::false\_type)`
- `template<typename _CharT, typename _Integer> const _CharT * \_\_gnu\_debug::\_\_check\_string (const _CharT * __s, const _Integer & __n \_\_attribute\_\_\(\(unused\)\))`
- `template<typename _CharT> const _CharT * \_\_gnu\_debug::\_\_check\_string (const _CharT * __s)`
- `template<typename _InputIterator> _InputIterator \_\_gnu\_debug::\_\_check\_valid\_range (const _InputIterator & __first, const _InputIterator & __last \_\_attribute\_\_\(\(unused\)\))`
- `template<typename _InputIterator> bool \_\_gnu\_debug::\_\_valid\_range (const _InputIterator & __first, const _InputIterator & __last)`
- `template<typename _Iterator, typename _Sequence> bool \_\_gnu\_debug::\_\_valid\_range (const _Safe_iterator< _Iterator, _Sequence> & __first, const _Safe_iterator< _Iterator, _Sequence> & __last)`
- `template<typename _Iterator, typename _Sequence> bool \_\_gnu\_debug::\_\_valid\_range (const _Safe_local_iterator< _Iterator, _Sequence> & __first, const _Safe_local_iterator< _Iterator, _Sequence> & __last)`
- `template<typename _Integral> bool \_\_gnu\_debug::\_\_valid\_range\_aux (const _Integral &, const _Integral &, std::true\_type)`
- `template<typename _InputIterator> bool \_\_gnu\_debug::\_\_valid\_range\_aux (const _InputIterator & __first, const _InputIterator & __last, std::false\_type)`

- `template<typename _RandomAccessIterator > bool __gnu_debug::__valid_range_aux2 (const _RandomAccessIterator &__first, const _RandomAccessIterator &__last, std::random\_access\_iterator\_tag)`
- `template<typename _InputIterator > bool __gnu_debug::__valid_range_aux2 (const _InputIterator &, const _InputIterator &, std::input\_iterator\_tag)`

#### 5.162.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

### 5.163 gp\_ht\_map.hpp File Reference

#### Classes

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

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_GEN_POS`
- `#define PB_DS_GP_HASH_NAME`
- `#define PB_DS_GP_HASH_TRAITS_BASE`
- `#define PB_DS_HASH_EQ_FN_C_DEC`
- `#define PB_DS_RANGED_PROBE_FN_C_DEC`

#### Variables

- `empty_entry_status`
- `erased_entry_status`
- `valid_entry_status`

#### 5.163.1 Detailed Description

Contains an implementation class for general probing hash.

### 5.164 gslice.h File Reference

#### Classes

- class [std::gslice](#)

## Namespaces

- [std](#)

### 5.164.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 5.165 `gslice_array.h` File Reference

### Classes

- class [std::gslice\\_array<\\_Tp>](#)

## Namespaces

- [std](#)

## Macros

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

### 5.165.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 5.166 `hash_bytes.h` File Reference

### Namespaces

- [std](#)

### Functions

- `size_t std::Fnv_hash_bytes(const void *__ptr, size_t __len, size_t __seed)`
- `size_t std::_Hash_bytes(const void *__ptr, size_t __len, size_t __seed)`

### 5.166.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 5.167 hash\_eq\_fn.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::hash\\_eq\\_fn](#)< Key, Eq\_Fn, \_Alloc, Store\_Hash >
- struct [\\_\\_gnu\\_pbds::detail::hash\\_eq\\_fn](#)< Key, Eq\_Fn, \_Alloc, false >
- struct [\\_\\_gnu\\_pbds::detail::hash\\_eq\\_fn](#)< Key, Eq\_Fn, \_Alloc, true >

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.167.1 Detailed Description

Contains 2 equivalence functions, one employing a hash value, and one ignoring it.

## 5.168 hash\_exponential\_size\_policy\_imp.hpp File Reference

#### 5.168.1 Detailed Description

Contains a resize size policy implementation.

## 5.169 hash\_fun.h File Reference

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### Functions

- `size_t __gnu_cxx::__stl_hash_string (const char *__s)`

#### 5.169.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 5.170 hash\_load\_check\_resize\_trigger\_imp.hpp File Reference

### Macros

- `#define PB_DS_ASSERT_VALID(X)`
- `#define PB_DS_ASSERT_VALID(X)`

#### 5.170.1 Detailed Description

Contains a resize trigger implementation.

## 5.171 hash\_load\_check\_resize\_trigger\_size\_base.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::hash\\_load\\_check\\_resize\\_trigger\\_size\\_base< Size\\_Type, Hold\\_Size >](#)
- class [\\_\\_gnu\\_pbds::detail::hash\\_load\\_check\\_resize\\_trigger\\_size\\_base< Size\\_Type, true >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### 5.171.1 Detailed Description

Contains an base holding size for some resize policies.

## 5.172 hash\_policy.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger< External\\_Load\\_Access, Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::direct\\_mask\\_range\\_hashing< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::direct\\_mod\\_range\\_hashing< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::hash\\_exponential\\_size\\_policy< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::hash\\_load\\_check\\_resize\\_trigger< External\\_Load\\_Access, Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::hash\\_prime\\_size\\_policy](#)
- class [\\_\\_gnu\\_pbds::hash\\_standard\\_resize\\_policy< Size\\_Policy, Trigger\\_Policy, External\\_Size\\_Access, Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::linear\\_probe\\_fn< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::quadratic\\_probe\\_fn< Size\\_Type >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`

- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_SIZE_BASE_C_DEC`

#### Enumerations

- `enum { num_distinct_sizes_32_bit, num_distinct_sizes_64_bit, num_distinct_sizes }`

#### Variables

- `static const std::size_t __gnu_pbds::detail::g_a_sizes [num_distinct_sizes_64_bit]`

#### 5.172.1 Detailed Description

Contains hash-related policies.

### 5.173 hash\_prime\_size\_policy\_imp.hpp File Reference

#### Enumerations

- `enum { num_distinct_sizes_32_bit, num_distinct_sizes_64_bit, num_distinct_sizes }`

#### Variables

- `static const std::size_t detail::g_a_sizes [num_distinct_sizes_64_bit]`

#### 5.173.1 Detailed Description

Contains a resize size policy implementation.

### 5.174 hash\_standard\_resize\_policy\_imp.hpp File Reference

#### 5.174.1 Detailed Description

Contains a resize policy implementation.

### 5.175 hashtable.h File Reference

#### Classes

- `class std::_Hashtable<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`

#### Namespaces

- `std`



## Typedefs

- `template<typename _Tp, typename _Hash> using std::__cache_default = __not_< __and_< __is_fast_hash< _↔  
Hash >, is_default_constructible< _Hash >, is_copy_assignable< _Hash >, __detail::__is_noexcept_hash<  
_Tp, _Hash >>>`

### 5.175.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_map>` or `<unordered_set>`.

## 5.176 hashtable.h File Reference

### Namespaces

- [`\_\_gnu\_cxx`](#)

### Enumerations

- `enum { _S_num_primes }`

### Functions

- `unsigned long __gnu_cxx::__stl_next_prime (unsigned long __n)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All> bool __gnu_cxx::operator!= (const  
hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All >  
&__ht2)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All> bool __gnu_cxx::operator== (const  
hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All >  
&__ht2)`
- `template<class _Val, class _Key, class _HF, class _Extract, class _EqKey, class _All> void __gnu_cxx::swap (hashtable<  
_Val, _Key, _HF, _Extract, _EqKey, _All > &__ht1, hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht2)`

### 5.176.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 5.177 hashtable\_policy.h File Reference

### Classes

- `struct std::__detail::Before_begin< _NodeAlloc >`
- `struct std::__detail::Default_ranged_hash`
- `struct std::__detail::Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, __cache_hash_code  
>`
- `struct std::__detail::Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, false >`
- `struct std::__detail::Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, true >`

- struct std::\_\_detail::Equality< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, Unique\_keys >
- struct std::\_\_detail::Equality< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, false >
- struct std::\_\_detail::Equality< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, true >
- struct std::\_\_detail::Equality\_base
- struct std::\_\_detail::Hash\_code\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_cache\_hash\_code >
- struct std::\_\_detail::Hash\_code\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, Default\_ranged\_hash, false >
- struct std::\_\_detail::Hash\_code\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, Default\_ranged\_hash, true >
- struct std::\_\_detail::Hash\_code\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, false >
- struct std::\_\_detail::Hash\_node< \_Value, \_Cache\_hash\_code >
- struct std::\_\_detail::Hash\_node< \_Value, false >
- struct std::\_\_detail::Hash\_node< \_Value, true >
- struct std::\_\_detail::Hash\_node\_base
- struct std::\_\_detail::Hashtable\_base< \_Key, \_Value, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_Traits >
- struct std::\_\_detail::Hashtable\_base< \_Key, \_Value, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_Traits >
- struct std::\_\_detail::Hashtable\_ebo\_helper< \_Nm, \_Tp, \_\_use\_ebo >
- struct std::\_\_detail::Hashtable\_ebo\_helper< \_Nm, \_Tp, false >
- struct std::\_\_detail::Hashtable\_ebo\_helper< \_Nm, \_Tp, true >
- struct std::\_\_detail::Hashtable\_traits< \_Cache\_hash\_code, \_Constant\_iterators, \_Unique\_keys >
- struct std::\_\_detail::Insert< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, \_Constant\_iterators, \_Unique\_keys >
- struct std::\_\_detail::Insert< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, false, \_Unique\_keys >
- struct std::\_\_detail::Insert< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, true, false >
- struct std::\_\_detail::Insert< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, true, true >
- struct std::\_\_detail::Insert\_base< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits >
- struct std::\_\_detail::Local\_const\_iterator< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_constant\_iterators, \_\_cache >
- struct std::\_\_detail::Local\_iterator< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_constant\_iterators, \_\_cache >
- struct std::\_\_detail::Local\_iterator\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_cache\_hash\_code >
- struct std::\_\_detail::Local\_iterator\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, false >
- struct std::\_\_detail::Local\_iterator\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, true >
- struct std::\_\_detail::Map\_base< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, \_Unique\_keys >
- struct std::\_\_detail::Map\_base< \_Key, \_Pair, \_Alloc, \_Select1st, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, false >
- struct std::\_\_detail::Map\_base< \_Key, \_Pair, \_Alloc, \_Select1st, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, true >
- struct std::\_\_detail::Mod\_range\_hashing
- struct std::\_\_detail::Node\_const\_iterator< \_Value, \_\_constant\_iterators, \_\_cache >
- struct std::\_\_detail::Node\_iterator< \_Value, \_\_constant\_iterators, \_\_cache >
- struct std::\_\_detail::Node\_iterator\_base< \_Value, \_Cache\_hash\_code >
- struct std::\_\_detail::Prime\_rehash\_policy
- struct std::\_\_detail::Rehash\_base< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits >
- struct std::\_\_detail::Rehash\_base< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_Prime\_rehash\_policy, \_Traits >
- class std::\_\_detail::Hashtable< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits >

## Namespaces

- [std](#)
- [std::\\_\\_detail](#)

## Functions

- `template<class _Iterator> std::iterator_traits<_Iterator>::difference_type std::__detail::__distance_fw (_Iterator __first, _Iterator __last, std::input\_iterator\_tag)`
- `template<class _Iterator> std::iterator_traits<_Iterator>::difference_type std::__detail::__distance_fw (_Iterator __first, _Iterator __last, std::forward\_iterator\_tag)`
- `template<class _Iterator> std::iterator_traits<_Iterator>::difference_type std::__detail::__distance_fw (_Iterator __first, _Iterator __last)`
- `template<typename _Value, bool _Cache_hash_code> bool std::__detail::operator!= (const _Node_iterator_base<_↵_Value, _Cache_hash_code> &__x, const _Node_iterator_base<_Value, _Cache_hash_code> &__y)`
- `template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache> bool std::__detail::operator!= (const _Local_iterator_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, _↵__cache> &__x, const _Local_iterator_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache> &__y)`
- `template<typename _Value, bool _Cache_hash_code> bool std::__detail::operator== (const _Node_iterator_base<_↵_Value, _Cache_hash_code> &__x, const _Node_iterator_base<_Value, _Cache_hash_code> &__y)`
- `template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache> bool std::__detail::operator== (const _Local_iterator_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, _↵__cache> &__x, const _Local_iterator_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache> &__y)`

## 5.177.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_map>` or `<unordered_set>`.

5.178 `indirect_array.h` File Reference

## Classes

- class [std::indirect\\_array<\\_Tp>](#)

## Namespaces

- [std](#)

## Macros

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

## 5.178.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 5.179 info\_fn\_imps.hpp File Reference

### 5.179.1 Detailed Description

Contains an implementation class for a binary\_heap.

## 5.180 info\_fn\_imps.hpp File Reference

### 5.180.1 Detailed Description

Contains an implementation class for bin\_search\_tree\_.

## 5.181 info\_fn\_imps.hpp File Reference

### 5.181.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s entire container info related functions.

## 5.182 info\_fn\_imps.hpp File Reference

### 5.182.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s entire container info related functions.

## 5.183 info\_fn\_imps.hpp File Reference

### 5.183.1 Detailed Description

Contains an implementation class for left\_child\_next\_sibling\_heap\_.

## 5.184 info\_fn\_imps.hpp File Reference

### 5.184.1 Detailed Description

Contains implementations of lu\_map\_.

## 5.185 info\_fn\_imps.hpp File Reference

### 5.185.1 Detailed Description

Contains an implementation class for ov\_tree\_.

## 5.186 info\_fn\_imps.hpp File Reference

### 5.186.1 Detailed Description

Contains an implementation class for pat\_trie.

**5.187 info\_fn\_imps.hpp File Reference****5.187.1 Detailed Description**

Contains an implementation for rb\_tree\_.

**5.188 info\_fn\_imps.hpp File Reference****5.188.1 Detailed Description**

Contains an implementation.

**5.189 insert\_fn\_imps.hpp File Reference****5.189.1 Detailed Description**

Contains an implementation class for a binary\_heap.

**5.190 insert\_fn\_imps.hpp File Reference****5.190.1 Detailed Description**

Contains an implementation class for a base of binomial heaps.

**5.191 insert\_fn\_imps.hpp File Reference****5.191.1 Detailed Description**

Contains an implementation class for bin\_search\_tree\_.

**5.192 insert\_fn\_imps.hpp File Reference****5.192.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s insert related functions.

**5.193 insert\_fn\_imps.hpp File Reference****5.193.1 Detailed Description**

Contains implementations of gp\_ht\_map\_'s insert related functions.

**5.194 insert\_fn\_imps.hpp File Reference****5.194.1 Detailed Description**

Contains an implementation class for left\_child\_next\_sibling\_heap\_.

## 5.195 insert\_fn\_imps.hpp File Reference

### 5.195.1 Detailed Description

Contains implementations of lu\_map\_.

## 5.196 insert\_fn\_imps.hpp File Reference

### 5.196.1 Detailed Description

Contains an implementation class for ov\_tree\_.

## 5.197 insert\_fn\_imps.hpp File Reference

### 5.197.1 Detailed Description

Contains an implementation class for a pairing heap.

## 5.198 insert\_fn\_imps.hpp File Reference

### 5.198.1 Detailed Description

Contains an implementation for rb\_tree\_.

## 5.199 insert\_fn\_imps.hpp File Reference

### 5.199.1 Detailed Description

Contains an implementation for rc\_binomial\_heap\_.

## 5.200 insert\_fn\_imps.hpp File Reference

### 5.200.1 Detailed Description

Contains an implementation class for splay\_tree\_.

## 5.201 insert\_fn\_imps.hpp File Reference

### 5.201.1 Detailed Description

Contains an implementation for thin\_heap\_.

## 5.202 insert\_join\_fn\_imps.hpp File Reference

### 5.202.1 Detailed Description

Contains an implementation class for pat\_trie.

### 5.203 `insert_no_store_hash_fn_imps.hpp` File Reference

#### 5.203.1 Detailed Description

Contains implementations of `cc_ht_map_'s` insert related functions, when the hash value is not stored.

### 5.204 `insert_no_store_hash_fn_imps.hpp` File Reference

#### 5.204.1 Detailed Description

Contains implementations of `gp_ht_map_'s` insert related functions, when the hash value is not stored.

### 5.205 `insert_store_hash_fn_imps.hpp` File Reference

#### 5.205.1 Detailed Description

Contains implementations of `cc_ht_map_'s` insert related functions, when the hash value is stored.

### 5.206 `insert_store_hash_fn_imps.hpp` File Reference

#### 5.206.1 Detailed Description

Contains implementations of `gp_ht_map_'s` find related functions, when the hash value is stored.

### 5.207 `ios_base.h` File Reference

#### Classes

- class [std::ios\\_base](#)
- class [std::ios\\_base::failure](#)

#### Namespaces

- [std](#)

#### Enumerations

- enum `_ios_Fmtflags` { `_S_boolalpha`, `_S_dec`, `_S_fixed`, `_S_hex`, `_S_internal`, `_S_left`, `_S_oct`, `_S_right`, `_S_scientific`, `_S_showbase`, `_S_showpoint`, `_S_showpos`, `_S_skipws`, `_S_unitbuf`, `_S_uppercase`, `_S_↔adjustfield`, `_S_basefield`, `_S_floatfield`, `_S_ios_fmtflags_end` }
- enum `_ios_Iostate` { `_S_goodbit`, `_S_badbit`, `_S_eofbit`, `_S_failbit`, `_S_ios_iostate_end` }
- enum `_ios_Openmode` { `_S_app`, `_S_ate`, `_S_bin`, `_S_in`, `_S_out`, `_S_trunc`, `_S_ios_openmode_end` }
- enum `_ios_Seekdir` { `_S_beg`, `_S_cur`, `_S_end`, `_S_ios_seekdir_end` }

## Functions

- ios\_base & [std::boolalpha](#) (ios\_base & \_\_base)
- ios\_base & [std::dec](#) (ios\_base & \_\_base)
- ios\_base & [std::fixed](#) (ios\_base & \_\_base)
- ios\_base & [std::hex](#) (ios\_base & \_\_base)
- ios\_base & [std::internal](#) (ios\_base & \_\_base)
- ios\_base & [std::left](#) (ios\_base & \_\_base)
- ios\_base & [std::noboolalpha](#) (ios\_base & \_\_base)
- ios\_base & [std::noshowbase](#) (ios\_base & \_\_base)
- ios\_base & [std::noshowpoint](#) (ios\_base & \_\_base)
- ios\_base & [std::noshowpos](#) (ios\_base & \_\_base)
- ios\_base & [std::noskipws](#) (ios\_base & \_\_base)
- ios\_base & [std::nounitbuf](#) (ios\_base & \_\_base)
- ios\_base & [std::nouppercase](#) (ios\_base & \_\_base)
- ios\_base & [std::oct](#) (ios\_base & \_\_base)
- constexpr \_ios\_Fmtflags **std::operator&** (\_ios\_Fmtflags \_\_a, \_ios\_Fmtflags \_\_b)
- constexpr \_ios\_Openmode **std::operator&** (\_ios\_Openmode \_\_a, \_ios\_Openmode \_\_b)
- constexpr \_ios\_istate **std::operator&** (\_ios\_istate \_\_a, \_ios\_istate \_\_b)
- const \_ios\_Fmtflags & **std::operator&=** (\_ios\_Fmtflags & \_\_a, \_ios\_Fmtflags \_\_b)
- const \_ios\_Openmode & **std::operator&=** (\_ios\_Openmode & \_\_a, \_ios\_Openmode \_\_b)
- const \_ios\_istate & **std::operator&=** (\_ios\_istate & \_\_a, \_ios\_istate \_\_b)
- constexpr \_ios\_Fmtflags **std::operator^** (\_ios\_Fmtflags \_\_a, \_ios\_Fmtflags \_\_b)
- constexpr \_ios\_Openmode **std::operator^** (\_ios\_Openmode \_\_a, \_ios\_Openmode \_\_b)
- constexpr \_ios\_istate **std::operator^** (\_ios\_istate \_\_a, \_ios\_istate \_\_b)
- const \_ios\_Fmtflags & **std::operator^=** (\_ios\_Fmtflags & \_\_a, \_ios\_Fmtflags \_\_b)
- const \_ios\_Openmode & **std::operator^=** (\_ios\_Openmode & \_\_a, \_ios\_Openmode \_\_b)
- const \_ios\_istate & **std::operator^=** (\_ios\_istate & \_\_a, \_ios\_istate \_\_b)
- constexpr \_ios\_Fmtflags **std::operator|** (\_ios\_Fmtflags \_\_a, \_ios\_Fmtflags \_\_b)
- constexpr \_ios\_Openmode **std::operator|** (\_ios\_Openmode \_\_a, \_ios\_Openmode \_\_b)
- constexpr \_ios\_istate **std::operator|** (\_ios\_istate \_\_a, \_ios\_istate \_\_b)
- const \_ios\_Fmtflags & **std::operator|=** (\_ios\_Fmtflags & \_\_a, \_ios\_Fmtflags \_\_b)
- const \_ios\_Openmode & **std::operator|=** (\_ios\_Openmode & \_\_a, \_ios\_Openmode \_\_b)
- const \_ios\_istate & **std::operator|=** (\_ios\_istate & \_\_a, \_ios\_istate \_\_b)
- constexpr \_ios\_Fmtflags **std::operator~** (\_ios\_Fmtflags \_\_a)
- constexpr \_ios\_Openmode **std::operator~** (\_ios\_Openmode \_\_a)
- constexpr \_ios\_istate **std::operator~** (\_ios\_istate \_\_a)
- ios\_base & [std::right](#) (ios\_base & \_\_base)
- ios\_base & [std::scientific](#) (ios\_base & \_\_base)
- ios\_base & [std::showbase](#) (ios\_base & \_\_base)
- ios\_base & [std::showpoint](#) (ios\_base & \_\_base)
- ios\_base & [std::showpos](#) (ios\_base & \_\_base)
- ios\_base & [std::skipws](#) (ios\_base & \_\_base)
- ios\_base & [std::unitbuf](#) (ios\_base & \_\_base)
- ios\_base & [std::uppercase](#) (ios\_base & \_\_base)

## 5.207.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.



## 5.208 iterator.h File Reference

### Classes

- class [\\_\\_gnu\\_parallel::\\_IteratorPair<\\_Iterator1, \\_Iterator2, \\_IteratorCategory >](#)
- class [\\_\\_gnu\\_parallel::\\_IteratorTriple<\\_Iterator1, \\_Iterator2, \\_Iterator3, \\_IteratorCategory >](#)

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### 5.208.1 Detailed Description

Helper iterator classes for the `std::transform()` functions. This file is a GNU parallel extension to the Standard C++ Library.

## 5.209 iterator.hpp File Reference

### Classes

- class [iterator\\_](#)

### 5.209.1 Detailed Description

Contains an `iterator_` class used for ranging over the elements of the table.

## 5.210 iterator\_fn\_imps.hpp File Reference

### 5.210.1 Detailed Description

Contains implementations of `gp_ht_map_'s` iterators related functions, e.g., `begin()`.

## 5.211 iterator\_tracker.h File Reference

### Namespaces

- [std](#)
- [std::\\_\\_profile](#)

### Functions

- `template<typename _IteratorL, typename _IteratorR, typename _Sequence > bool std::__profile::operator!= (const __↵  
iterator_tracker<_IteratorL, _Sequence > &__lhs, const __iterator_tracker<_IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence > bool std::__profile::operator!= (const __iterator_tracker<_↵  
Iterator, _Sequence > &__lhs, const __iterator_tracker<_Iterator, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence > __iterator_tracker<_Iterator, _Sequence > std::__profile↵  
::operator+ (typename __iterator_tracker<_Iterator, _Sequence >::difference_type __n, const __iterator_↵  
tracker<_Iterator, _Sequence > &__i)`

- `template<typename _IteratorL, typename _IteratorR, typename _Sequence> __iterator_tracker<_IteratorL, _Sequence> <::difference_type std::__profile::operator- (const __iterator_tracker<_IteratorL, _Sequence> &__lhs, const __iterator_tracker<_IteratorR, _Sequence> &__rhs)`
- `template<typename _Iterator, typename _Sequence> __iterator_tracker<_Iterator, _Sequence> <::difference_type std::__profile::operator- (const __iterator_tracker<_Iterator, _Sequence> &__lhs, const __iterator_tracker<_Iterator, _Sequence> &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence> bool std::__profile::operator< (const __iterator_tracker<_IteratorL, _Sequence> &__lhs, const __iterator_tracker<_IteratorR, _Sequence> &__rhs)`
- `template<typename _Iterator, typename _Sequence> bool std::__profile::operator< (const __iterator_tracker<_Iterator, _Sequence> &__lhs, const __iterator_tracker<_Iterator, _Sequence> &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence> bool std::__profile::operator<= (const __iterator_tracker<_IteratorL, _Sequence> &__lhs, const __iterator_tracker<_IteratorR, _Sequence> &__rhs)`
- `template<typename _Iterator, typename _Sequence> bool std::__profile::operator<= (const __iterator_tracker<_Iterator, _Sequence> &__lhs, const __iterator_tracker<_Iterator, _Sequence> &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence> bool std::__profile::operator== (const __iterator_tracker<_IteratorL, _Sequence> &__lhs, const __iterator_tracker<_IteratorR, _Sequence> &__rhs)`
- `template<typename _Iterator, typename _Sequence> bool std::__profile::operator== (const __iterator_tracker<_Iterator, _Sequence> &__lhs, const __iterator_tracker<_Iterator, _Sequence> &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence> bool std::__profile::operator> (const __iterator_tracker<_IteratorL, _Sequence> &__lhs, const __iterator_tracker<_IteratorR, _Sequence> &__rhs)`
- `template<typename _Iterator, typename _Sequence> bool std::__profile::operator> (const __iterator_tracker<_Iterator, _Sequence> &__lhs, const __iterator_tracker<_Iterator, _Sequence> &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence> bool std::__profile::operator>= (const __iterator_tracker<_IteratorL, _Sequence> &__lhs, const __iterator_tracker<_IteratorR, _Sequence> &__rhs)`
- `template<typename _Iterator, typename _Sequence> bool std::__profile::operator>= (const __iterator_tracker<_Iterator, _Sequence> &__lhs, const __iterator_tracker<_Iterator, _Sequence> &__rhs)`

#### 5.211.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

## 5.212 iterators\_fn\_imps.hpp File Reference

#### 5.212.1 Detailed Description

Contains an implementation class for a binary\_heap.

## 5.213 iterators\_fn\_imps.hpp File Reference

#### 5.213.1 Detailed Description

Contains an implementation class for bin\_search\_tree\_.

## 5.214 iterators\_fn\_imps.hpp File Reference

#### 5.214.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s iterators related functions, e.g., begin().

## 5.215 iterators\_fn\_imps.hpp File Reference

### 5.215.1 Detailed Description

Contains an implementation class for left\_child\_next\_sibling\_heap\_.

## 5.216 iterators\_fn\_imps.hpp File Reference

### 5.216.1 Detailed Description

Contains implementations of lu\_map\_.

## 5.217 iterators\_fn\_imps.hpp File Reference

### 5.217.1 Detailed Description

Contains an implementation class for ov\_tree\_.

## 5.218 iterators\_fn\_imps.hpp File Reference

### 5.218.1 Detailed Description

Contains an implementation class for pat\_trie.

## 5.219 left\_child\_next\_sibling\_heap\_.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::left\\_child\\_next\\_sibling\\_heap< Value\\_Type, Cmp\\_Fn, Node\\_Metadata, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

### 5.219.1 Detailed Description

Contains an implementation class for a basic heap.

## 5.220 linear\_probe\_fn\_imp.hpp File Reference

### 5.220.1 Detailed Description

Contains a probe policy implementation

## 5.221 list\_partition.h File Reference

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _Iter > void \_\_gnu\_parallel::\_\_shrink (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_length)`
- `template<typename _Iter > void \_\_gnu\_parallel::\_\_shrink\_and\_double (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_length, const bool __make_twice)`
- `template<typename _Iter, typename _FunctorType > size_t \_\_gnu\_parallel::list\_partition (const _Iter __begin, const _Iter __end, _Iter * __starts, size_t * __lengths, const int __num_parts, _FunctorType & __f, int __oversampling=0)`

#### 5.221.1 Detailed Description

\_\_Functionality to split \_\_sequence referenced by only input iterators. This file is a GNU parallel extension to the Standard C++ Library.

## 5.222 list\_update\_policy.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::lu\\_counter\\_policy< Max\\_Count, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::lu\\_move\\_to\\_front\\_policy< \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.222.1 Detailed Description

Contains policies for list update containers.

## 5.223 locale\_classes.h File Reference

### Classes

- class [std::collate< \\_CharT >](#)
- class [std::collate\\_byname< \\_CharT >](#)
- class [std::locale](#)
- class [std::locale::facet](#)
- class [std::locale::id](#)

### Namespaces

- [std](#)

### 5.223.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 5.224 `locale_facets.h` File Reference

### Classes

- class `std::__ctype_abstract_base<_CharT>`
- class `std::ctype<_CharT>`
- class `std::ctype<char>`
- class `std::ctype<wchar_t>`
- class `std::ctype_byname<_CharT>`
- class `std::ctype_byname<char>`
- class `std::num_get<_CharT, _Inlter>`
- class `std::num_put<_CharT, _Outlter>`
- class `std::num_punct<_CharT>`
- class `std::num_punct_byname<_CharT>`

### Namespaces

- `std`

### Macros

- `#define _GLIBCXX_NUM_FACETS`

### Functions

- `template<typename _CharT> _CharT * std::__add_grouping (_CharT *__s, _CharT __sep, const char *__gbeg, size_t __gsize, const _CharT *__first, const _CharT *__last)`
- `template<typename _Tp> void std::__convert_to_v (const char *, _Tp &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<> void std::__convert_to_v (const char *, float &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<> void std::__convert_to_v (const char *, double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<> void std::__convert_to_v (const char *, long double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<typename _CharT> ostreambuf_iterator<_CharT> std::__write (ostreambuf_iterator<_CharT> __s, const _CharT *__ws, int __len)`
- `template<typename _CharT, typename _Outlter> _Outlter std::__write (_Outlter __s, const _CharT *__ws, int __len)`
- `template<typename _CharT> bool std::isalnum (_CharT __c, const locale &__loc)`
- `template<typename _CharT> bool std::isalpha (_CharT __c, const locale &__loc)`
- `template<typename _CharT> bool std::iscntrl (_CharT __c, const locale &__loc)`
- `template<typename _CharT> bool std::isdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT> bool std::isgraph (_CharT __c, const locale &__loc)`
- `template<typename _CharT> bool std::islower (_CharT __c, const locale &__loc)`
- `template<typename _CharT> bool std::isprint (_CharT __c, const locale &__loc)`
- `template<typename _CharT> bool std::ispunct (_CharT __c, const locale &__loc)`
- `template<typename _CharT> bool std::isspace (_CharT __c, const locale &__loc)`

- template<typename \_CharT> bool [std::isupper](#) (\_CharT \_\_c, const locale &\_\_loc)
- template<typename \_CharT> bool [std::isxdigit](#) (\_CharT \_\_c, const locale &\_\_loc)
- template<typename \_CharT> \_CharT [std::tolower](#) (\_CharT \_\_c, const locale &\_\_loc)
- template<typename \_CharT> \_CharT [std::toupper](#) (\_CharT \_\_c, const locale &\_\_loc)

#### 5.224.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 5.225 locale\_facets\_nonio.h File Reference

### Classes

- class [std::messages< \\_CharT >](#)
- struct [std::messages\\_base](#)
- class [std::messages\\_byname< \\_CharT >](#)
- class [std::money\\_base](#)
- class [std::money\\_get< \\_CharT, \\_InIter >](#)
- class [std::money\\_put< \\_CharT, \\_OutIter >](#)
- class [std::moneypunct< \\_CharT, \\_Intl >](#)
- class [std::moneypunct\\_byname< \\_CharT, \\_Intl >](#)
- class [std::time\\_base](#)
- class [std::time\\_get< \\_CharT, \\_InIter >](#)
- class [std::time\\_get\\_byname< \\_CharT, \\_InIter >](#)
- class [std::time\\_put< \\_CharT, \\_OutIter >](#)
- class [std::time\\_put\\_byname< \\_CharT, \\_OutIter >](#)

### Namespaces

- [std](#)

#### 5.225.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 5.226 localefwd.h File Reference

### Classes

- class [std::codecvt< \\_InternT, \\_ExternT, \\_StateT >](#)
- class [std::codecvt\\_byname< \\_InternT, \\_ExternT, \\_StateT >](#)
- class [std::collate< \\_CharT >](#)
- class [std::collate\\_byname< \\_CharT >](#)
- class [std::ctype< \\_CharT >](#)
- class [std::ctype\\_byname< \\_CharT >](#)
- class [std::messages< \\_CharT >](#)
- class [std::messages\\_byname< \\_CharT >](#)

- class `std::money_get<_CharT, _InIter >`
- class `std::money_put<_CharT, _OutIter >`
- class `std::moneypunct<_CharT, _Intl >`
- class `std::moneypunct_byname<_CharT, _Intl >`
- class `std::num_get<_CharT, _InIter >`
- class `std::num_put<_CharT, _OutIter >`
- class `std::numpunct<_CharT >`
- class `std::numpunct_byname<_CharT >`
- class `std::time_get<_CharT, _InIter >`
- class `std::time_get_byname<_CharT, _InIter >`
- class `std::time_put<_CharT, _OutIter >`
- class `std::time_put_byname<_CharT, _OutIter >`

## Namespaces

- `std`

## Functions

- `template<typename _Facet > bool std::has_facet (const locale &) throw ()`
- `template<typename _CharT > bool std::isalnum (_CharT __c, const locale &__loc)`
- `template<typename _CharT > bool std::isalpha (_CharT __c, const locale &__loc)`
- `template<typename _CharT > bool std::isctrl (_CharT __c, const locale &__loc)`
- `template<typename _CharT > bool std::isdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT > bool std::isgraph (_CharT __c, const locale &__loc)`
- `template<typename _CharT > bool std::islower (_CharT __c, const locale &__loc)`
- `template<typename _CharT > bool std::isprint (_CharT __c, const locale &__loc)`
- `template<typename _CharT > bool std::ispunct (_CharT __c, const locale &__loc)`
- `template<typename _CharT > bool std::isspace (_CharT __c, const locale &__loc)`
- `template<typename _CharT > bool std::isupper (_CharT __c, const locale &__loc)`
- `template<typename _CharT > bool std::isxdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT > _CharT std::tolower (_CharT __c, const locale &__loc)`
- `template<typename _CharT > _CharT std::toupper (_CharT __c, const locale &__loc)`
- `template<typename _Facet > const _Facet & std::use_facet (const locale &)`

### 5.226.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

### 5.227 losertree.h File Reference

#### Classes

- class `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >`
- class `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`
- class `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >`
- struct `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser`
- class `__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >`

- class [\\_\\_gnu\\_parallel::\\_LoserTreePointer< false, \\_Tp, \\_Compare >](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreePointerBase< \\_Tp, \\_Compare >](#)
- struct [\\_\\_gnu\\_parallel::\\_LoserTreePointerBase< \\_Tp, \\_Compare >::\\_Loser](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreePointerUnguarded< \\_\\_stable, \\_Tp, \\_Compare >](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreePointerUnguarded< false, \\_Tp, \\_Compare >](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreePointerUnguardedBase< \\_Tp, \\_Compare >](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreeUnguarded< \\_\\_stable, \\_Tp, \\_Compare >](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreeUnguarded< false, \\_Tp, \\_Compare >](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreeUnguardedBase< \\_Tp, \\_Compare >](#)

#### Namespaces

- [\\_\\_gnu\\_parallel](#)

##### 5.227.1 Detailed Description

Many generic loser tree variants. This file is a GNU parallel extension to the Standard C++ Library.

## 5.228 lu\_counter\_metadata.hpp File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::detail::lu\\_counter\\_metadata< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::detail::lu\\_counter\\_policy\\_base< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::detail::lu\\_counter\\_policy\\_base< Size\\_Type >](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

##### 5.228.1 Detailed Description

Contains implementation of a lu counter policy's metadata.

## 5.229 lu\_map.hpp File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::detail::lu\\_map< Key, Mapped, Eq\\_Fn, \\_Alloc, Update\\_Policy >](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)



## Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_GEN_POS`
- `#define PB_DS_LU_NAME`
- `#define PB_DS_LU_TRAITS_BASE`

### 5.229.1 Detailed Description

Contains a list update map.

## 5.230 macros.h File Reference

### Macros

- `#define __glibcxx_check_bucket_index(_N)`
- `#define __glibcxx_check_equal_allocs(_Other)`
- `#define __glibcxx_check_erase(_Position)`
- `#define __glibcxx_check_erase_after(_Position)`
- `#define __glibcxx_check_erase_range(_First, _Last)`
- `#define __glibcxx_check_erase_range_after(_First, _Last)`
- `#define __glibcxx_check_heap(_First, _Last)`
- `#define __glibcxx_check_heap_pred(_First, _Last, _Pred)`
- `#define __glibcxx_check_insert(_Position)`
- `#define __glibcxx_check_insert_after(_Position)`
- `#define __glibcxx_check_insert_range(_Position, _First, _Last)`
- `#define __glibcxx_check_insert_range_after(_Position, _First, _Last)`
- `#define __glibcxx_check_max_load_factor(_F)`
- `#define __glibcxx_check_non_empty_range(_First, _Last)`
- `#define __glibcxx_check_nonempty()`
- `#define __glibcxx_check_partitioned_lower(_First, _Last, _Value)`
- `#define __glibcxx_check_partitioned_lower_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_check_partitioned_upper(_First, _Last, _Value)`
- `#define __glibcxx_check_partitioned_upper_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_check_self_move_assign(_Other)`
- `#define __glibcxx_check_sorted(_First, _Last)`
- `#define __glibcxx_check_sorted_pred(_First, _Last, _Pred)`
- `#define __glibcxx_check_sorted_set(_First1, _Last1, _First2)`
- `#define __glibcxx_check_sorted_set_pred(_First1, _Last1, _First2, _Pred)`
- `#define __glibcxx_check_string(_String)`
- `#define __glibcxx_check_string_len(_String, _Len)`
- `#define __glibcxx_check_subscript(_N)`
- `#define __glibcxx_check_valid_range(_First, _Last)`
- `#define __GLIBCXX_DEBUG_VERIFY(_Condition, _ErrorMessage)`
- `#define __GLIBCXX_DEBUG_VERIFY_AT(_Condition, _ErrorMessage, _File, _Line)`

### 5.230.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.230.2 Macro Definition Documentation

5.230.2.1 `#define __glibcxx_check_erase( _Position )`

Verify that we can erase the element referenced by the iterator `_Position`. We can erase the element if the `_Position` iterator is dereferenceable and references this sequence.

Definition at line 135 of file `macros.h`.

5.230.2.2 `#define __glibcxx_check_erase_after( _Position )`

Verify that we can erase the element after the iterator `_Position`. We can erase the element if the `_Position` iterator is before a dereferenceable one and references this sequence.

Definition at line 149 of file `macros.h`.

5.230.2.3 `#define __glibcxx_check_erase_range( _First, _Last )`

Verify that we can erase the elements in the iterator range `[_First, _Last)`. We can erase the elements if `[_First, _Last)` is a valid iterator range within this sequence.

Definition at line 163 of file `macros.h`.

5.230.2.4 `#define __glibcxx_check_erase_range_after( _First, _Last )`

Verify that we can erase the elements in the iterator range `[_First, _Last)`. We can erase the elements if `[_First, _Last)` is a valid iterator range within this sequence.

Definition at line 175 of file `macros.h`.

5.230.2.5 `#define __glibcxx_check_heap_pred( _First, _Last, _Pred )`

Verify that the iterator range `[_First, _Last)` is a heap w.r.t. the predicate `_Pred`.

Definition at line 318 of file `macros.h`.

5.230.2.6 `#define __glibcxx_check_insert( _Position )`

Verify that we can insert into `*this` with the iterator `_Position`. Insertion into a container at a specific position requires that the iterator be nonsingular, either dereferenceable or past-the-end, and that it reference the sequence we are inserting into. Note that this macro is only valid when the container is a `_Safe_sequence` and the iterator is a `_Safe_iterator`.

Definition at line 73 of file `macros.h`.

5.230.2.7 `#define __glibcxx_check_insert_after( _Position )`

Verify that we can insert into `*this` after the iterator `_Position`. Insertion into a container after a specific position requires that the iterator be nonsingular, either dereferenceable or before-begin, and that it reference the sequence we are inserting into. Note that this macro is only valid when the container is a `_Safe_sequence` and the iterator is a `_Safe_iterator`.

Definition at line 90 of file `macros.h`.

5.230.2.8 `#define __glibcxx_check_insert_range( _Position, _First, _Last )`

Verify that we can insert the values in the iterator range `[_First, _Last)` into `*this` with the iterator `_Position`. Insertion into a container at a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end), that it reference the sequence we are inserting into, and that the iterator range `[_First, _Last)` is a valid (possibly empty) range. Note that this macro is only valid when the container is a `_Safe_sequence` and the iterator is a `_Safe_iterator`.

**Todo** We would like to be able to check for noninterference of `_Position` and the range `[_First, _Last)`, but that can't (in general) be done.

Definition at line 110 of file `macros.h`.

5.230.2.9 `#define __glibcxx_check_insert_range_after( _Position, _First, _Last )`

Verify that we can insert the values in the iterator range `[_First, _Last)` into `*this` after the iterator `_Position`. Insertion into a container after a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end), that it reference the sequence we are inserting into, and that the iterator range `[_First, _Last)` is a valid (possibly empty) range. Note that this macro is only valid when the container is a `_Safe_sequence` and the iterator is a `_Safe_iterator`.

**Todo** We would like to be able to check for noninterference of `_Position` and the range `[_First, _Last)`, but that can't (in general) be done.

Definition at line 127 of file `macros.h`.

5.230.2.10 `#define __glibcxx_check_partitioned_lower( _First, _Last, _Value )`

Verify that the iterator range `[_First, _Last)` is partitioned w.r.t. the value `_Value`.

Definition at line 262 of file `macros.h`.

5.230.2.11 `#define __glibcxx_check_partitioned_lower_pred( _First, _Last, _Value, _Pred )`

Verify that the iterator range `[_First, _Last)` is partitioned w.r.t. the value `_Value` and predicate `_Pred`.

Definition at line 284 of file `macros.h`.

5.230.2.12 `#define __glibcxx_check_partitioned_upper_pred( _First, _Last, _Value, _Pred )`

Verify that the iterator range `[_First, _Last)` is partitioned w.r.t. the value `_Value` and predicate `_Pred`.

Definition at line 297 of file `macros.h`.

5.230.2.13 `#define __glibcxx_check_sorted_pred( _First, _Last, _Pred )`

Verify that the iterator range `[_First, _Last)` is sorted by the predicate `_Pred`.

Definition at line 233 of file `macros.h`.

5.230.2.14 `#define _GLIBCXX_DEBUG_VERIFY_AT( _Condition, _ErrorMessage, _File, _Line )`

Macros used by the implementation to verify certain properties. These macros may only be used directly by the debug wrappers. Note that these are macros (instead of the more obviously *correct* choice of making them functions) because we need line and file information at the call site, to minimize the distance between the user error and where the error is reported.

Definition at line 41 of file `macros.h`.

## 5.231 `malloc_allocator.h` File Reference

### Classes

- class `__gnu_cxx::malloc_allocator< _Tp >`

## Namespaces

- [\\_\\_gnu\\_cxx](#)

## Functions

- `template<typename _Tp> bool \_\_gnu\_cxx::operator!= (const malloc_allocator< _Tp > &, const malloc_allocator< _Tp > &)`
- `template<typename _Tp> bool \_\_gnu\_cxx::operator== (const malloc_allocator< _Tp > &, const malloc_allocator< _Tp > &)`

## 5.231.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.232 map.h File Reference

## Classes

- `class std::\_\_debug::map< _Key, _Tp, _Compare, _Allocator >`

## Namespaces

- [std](#)
- [std::\\_\\_debug](#)

## Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> bool std::\_\_debug::operator!= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> bool std::\_\_debug::operator< (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> bool std::\_\_debug::operator<= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> bool std::\_\_debug::operator== (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> bool std::\_\_debug::operator> (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> bool std::\_\_debug::operator>= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> void std::\_\_debug::swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare, _Allocator > &__rhs)`

## 5.232.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.233 map.h File Reference

### Classes

- class [std::\\_\\_profile::map<\\_Key, \\_Tp, \\_Compare, \\_Allocator>](#)

### Namespaces

- [std](#)
- [std::\\_\\_profile](#)

### Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> bool std::__profile::operator!= (const map<_Key, _Tp, _Compare, _Allocator> &__lhs, const map<_Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> bool std::__profile::operator< (const map<_Key, _Tp, _Compare, _Allocator> &__lhs, const map<_Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> bool std::__profile::operator<= (const map<_Key, _Tp, _Compare, _Allocator> &__lhs, const map<_Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> bool std::__profile::operator== (const map<_Key, _Tp, _Compare, _Allocator> &__lhs, const map<_Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> bool std::__profile::operator> (const map<_Key, _Tp, _Compare, _Allocator> &__lhs, const map<_Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> bool std::__profile::operator>= (const map<_Key, _Tp, _Compare, _Allocator> &__lhs, const map<_Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> void std::__profile::swap (map<_Key, _Tp, _Compare, _Allocator> &__lhs, map<_Key, _Tp, _Compare, _Allocator> &__rhs)`

### 5.233.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

## 5.234 mask\_array.h File Reference

### Classes

- class [std::mask\\_array<\\_Tp>](#)

### Namespaces

- [std](#)

### Macros

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

### 5.234.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 5.235 mask\_based\_range\_hashing.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::mask\\_based\\_range\\_hashing< Size\\_Type >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.235.1 Detailed Description

Contains a range hashing policy base.

## 5.236 memoryfwd.h File Reference

### Classes

- class [std::allocator< \\_Tp >](#)
- struct [std::uses\\_allocator< \\_Tp, \\_Alloc >](#)

### Namespaces

- [std](#)

#### 5.236.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 5.237 merge.h File Reference

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Functions

- [template<typename \\_RAIter1, typename \\_RAIter2, typename \\_OutputIterator, typename \\_DifferenceTp, typename \\_Compare > ↵↵ OutputIterator \\_\\_gnu\\_parallel::\\_\\_merge\\_advance \(\\_RAIter1 &\\_\\_begin1, \\_RAIter1 \\_\\_end1, \\_RAIter2 &\\_\\_begin2, \\_RAIter2 \\_\\_end2, \\_OutputIterator \\_\\_target, \\_DifferenceTp \\_\\_max\\_length, \\_Compare \\_\\_comp\)](#)
- [template<typename \\_RAIter1, typename \\_RAIter2, typename \\_OutputIterator, typename \\_DifferenceTp, typename \\_Compare > ↵↵ OutputIterator \\_\\_gnu\\_parallel::\\_\\_merge\\_advance\\_movc \(\\_RAIter1 &\\_\\_begin1, \\_RAIter1 \\_\\_end1, \\_RAIter2 &↵↵ \\_\\_begin2, \\_RAIter2 \\_\\_end2, \\_OutputIterator \\_\\_target, \\_DifferenceTp \\_\\_max\\_length, \\_Compare \\_\\_comp\)](#)
- [template<typename \\_RAIter1, typename \\_RAIter2, typename \\_OutputIterator, typename \\_DifferenceTp, typename \\_Compare > ↵↵ OutputIterator \\_\\_gnu\\_parallel::\\_\\_merge\\_advance\\_usual \(\\_RAIter1 &\\_\\_begin1, \\_RAIter1 \\_\\_end1, \\_RAIter2 &↵↵ \\_\\_begin2, \\_RAIter2 \\_\\_end2, \\_OutputIterator \\_\\_target, \\_DifferenceTp \\_\\_max\\_length, \\_Compare \\_\\_comp\)](#)

- `template<typename _RAIter1 , typename _RAIter2 , typename _RAIter3 , typename _Compare > _RAIter3 \_\_gnu\_parallel::\_\_parallel\_merge\_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _RAIter3 __target, typename std::iterator_traits< _RAIter1 >::difference_type __max_length, _Compare __comp)`
- `template<typename _RAIter1 , typename _RAIter3 , typename _Compare > _RAIter3 \_\_gnu\_parallel::\_\_parallel\_merge\_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter1 &__begin2, _RAIter1 __end2, _RAIter3 __target, typename std::iterator_traits< _RAIter1 >::difference_type __max_length, _Compare __comp)`

#### 5.237.1 Detailed Description

Parallel implementation of `std::merge()`. This file is a GNU parallel extension to the Standard C++ Library.

### 5.238 `messages_members.h` File Reference

#### Namespaces

- [std](#)

#### 5.238.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

### 5.239 `mod_based_range_hashing.hpp` File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::detail::mod\\_based\\_range\\_hashing< Size\\_Type >](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.239.1 Detailed Description

Contains a range hashing policy base.

### 5.240 `move.h` File Reference

#### Namespaces

- [std](#)

#### Macros

- `#define \_GLIBCXX\_FORWARD(_Tp, __val)`
- `#define \_GLIBCXX\_MOVE(__val)`

## Functions

- `template<typename _Tp> _Tp * std::\_\_addressof (_Tp &__r) noexcept`
- `template<typename _Tp> _Tp * std::addressof (_Tp &__r) noexcept`
- `template<typename _Tp> constexpr _Tp && std::forward (typename std::remove_reference< _Tp >::type &__t) noexcept`
- `template<typename _Tp> constexpr _Tp && std::forward (typename std::remove_reference< _Tp >::type &&__t) noexcept`
- `template<typename _Tp> constexpr std::remove_reference< _Tp >::type && std::move (_Tp &&__t) noexcept`
- `template<typename _Tp> constexpr conditional< __move_if_noexcept_cond< _Tp >::value, const _Tp &, _Tp && >::type std::move\_if\_noexcept (_Tp &__x) noexcept`
- `template<typename _Tp> void std::swap (_Tp &__a, _Tp &__b) noexcept(__and_< is_nothrow_move_constructible< _Tp >, is_nothrow_move_assignable< _Tp > >::value)`
- `template<typename _Tp, size_t _Nm> void std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(noexcept(swap(*__a, *__b)))`

## 5.240.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

## 5.241 mt\_allocator.h File Reference

## Classes

- `struct \_\_gnu\_cxx::\_\_common\_pool\_policy< _PoolTp, _Thread >`
- `class \_\_gnu\_cxx::\_\_mt\_alloc< _Tp, _Poolp >`
- `class \_\_gnu\_cxx::\_\_mt\_alloc\_base< _Tp >`
- `struct \_\_gnu\_cxx::\_\_per\_type\_pool\_policy< _Tp, _PoolTp, _Thread >`
- `class \_\_gnu\_cxx::\_\_pool< _Thread >`
- `class \_\_gnu\_cxx::\_\_pool< false >`
- `class \_\_gnu\_cxx::\_\_pool< true >`
- `struct \_\_gnu\_cxx::\_\_pool\_base`

## Namespaces

- [\\_\\_gnu\\_cxx](#)

## Macros

- `#define \_\_thread\_default`

## Typedefs

- `typedef void(* \_\_gnu\_cxx::\_\_destroy\_handler) (void *)`



## Functions

- `template<typename _Tp, typename _Poolp > bool __gnu_cxx::operator!= (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)`
- `template<typename _Tp, typename _Poolp > bool __gnu_cxx::operator== (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)`

### 5.241.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.242 `multimap.h` File Reference

### Classes

- class `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >`

### Namespaces

- `std`
- `std::__debug`

## Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator > bool std::__debug::operator!= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator > bool std::__debug::operator< (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator > bool std::__debug::operator<= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator > bool std::__debug::operator== (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator > bool std::__debug::operator> (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator > bool std::__debug::operator>= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator > void std::__debug::swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`

### 5.242.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

5.243 `multimap.h` File Reference

## Classes

- class `std::__profile::multimap<_Key, _Tp, _Compare, _Allocator>`

## Namespaces

- `std`
- `std::__profile`

## Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> bool std::__profile::operator!= (const multimap<_Key, _Tp, _Compare, _Allocator> &__lhs, const multimap<_Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> bool std::__profile::operator< (const multimap<_Key, _Tp, _Compare, _Allocator> &__lhs, const multimap<_Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> bool std::__profile::operator<= (const multimap<_Key, _Tp, _Compare, _Allocator> &__lhs, const multimap<_Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> bool std::__profile::operator== (const multimap<_Key, _Tp, _Compare, _Allocator> &__lhs, const multimap<_Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> bool std::__profile::operator> (const multimap<_Key, _Tp, _Compare, _Allocator> &__lhs, const multimap<_Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> bool std::__profile::operator>= (const multimap<_Key, _Tp, _Compare, _Allocator> &__lhs, const multimap<_Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator> void std::__profile::swap (multimap<_Key, _Tp, _Compare, _Allocator> &__lhs, multimap<_Key, _Tp, _Compare, _Allocator> &__rhs)`

## 5.243.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

5.244 `multiseq_selection.h` File Reference

## Classes

- class `__gnu_parallel::_Lexicographic<_T1, _T2, _Compare>`
- class `__gnu_parallel::_LexicographicReverse<_T1, _T2, _Compare>`

## Namespaces

- `__gnu_parallel`

## Macros

- `#define __S(__i)`
- `#define __S(__i)`

## Functions

- `template<typename _RanSeqs , typename _RankType , typename _RankIterator , typename _Compare > void __gnu_parallel::multiseq_partition ( _RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankIterator __begin_offsets, _Compare __comp=std::less< typename std::iterator_traits< typename std::iterator_traits< _RanSeqs >::value_type::first_type >::value_type >() )`
- `template<typename _Tp , typename _RanSeqs , typename _RankType , typename _Compare > _Tp __gnu_parallel::multiseq_selection ( _RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankType &__offset, _Compare __comp=std::less< _Tp >() )`

### 5.244.1 Detailed Description

Functions to find elements of a certain global `__rank` in multiple sorted sequences. Also serves for splitting such sequence sets.

The algorithm description can be found in

P. J. Varman, S. D. Scheufler, B. R. Iyer, and G. R. Ricard. Merging Multiple Lists on Hierarchical-Memory Multiprocessors. *Journal of Parallel and Distributed Computing*, 12(2):171–177, 1991.

This file is a GNU parallel extension to the Standard C++ Library.

### 5.245 multiset.h File Reference

#### Classes

- `class std::__debug::multiset< _Key, _Compare, _Allocator >`

#### Namespaces

- `std`
- `std::__debug`

#### Functions

- `template<typename _Key , typename _Compare , typename _Allocator > bool std::__debug::operator!= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator > bool std::__debug::operator< (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator > bool std::__debug::operator<= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator > bool std::__debug::operator== (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator > bool std::__debug::operator> (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator > bool std::__debug::operator>= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`

- `template<typename _Key , typename _Compare , typename _Allocator > void std::__debug::swap (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__y)`

#### 5.245.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.246 multiset.h File Reference

### Classes

- `class std::__profile::multiset< _Key, _Compare, _Allocator >`

### Namespaces

- `std`
- `std::__profile`

### Functions

- `template<typename _Key , typename _Compare , typename _Allocator > bool std::__profile::operator!= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator > bool std::__profile::operator< (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator > bool std::__profile::operator<= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator > bool std::__profile::operator== (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator > bool std::__profile::operator> (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator > bool std::__profile::operator>= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator > void std::__profile::swap (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__y)`

#### 5.246.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

## 5.247 multiway\_merge.h File Reference

### Classes

- `struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, __DifferenceTp, _Compare >`
- `struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, __DifferenceTp, _Compare >`
- `struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, __DifferenceTp, _Compare >`

- struct [\\_\\_gnu\\_parallel::\\_\\_multiway\\_merge\\_4\\_variant\\_sentinel\\_switch](#)< true, [\\_RAIterIterator](#), [\\_RAIter3](#), [\\_RAIter3\\_DifferenceTp](#), [\\_Compare](#) >
- struct [\\_\\_gnu\\_parallel::\\_\\_multiway\\_merge\\_k\\_variant\\_sentinel\\_switch](#)< [\\_\\_sentinels](#), [\\_\\_stable](#), [\\_RAIterIterator](#), [\\_RAIter3\\_DifferenceTp](#), [\\_Compare](#) >
- struct [\\_\\_gnu\\_parallel::\\_\\_multiway\\_merge\\_k\\_variant\\_sentinel\\_switch](#)< false, [\\_\\_stable](#), [\\_RAIterIterator](#), [\\_RAIter3\\_DifferenceTp](#), [\\_Compare](#) >
- class [\\_\\_gnu\\_parallel::\\_GuardedIterator](#)< [\\_RAIter](#), [\\_Compare](#) >
- struct [\\_\\_gnu\\_parallel::\\_LoserTreeTraits](#)< [\\_Tp](#) >
- struct [\\_\\_gnu\\_parallel::\\_SamplingSorter](#)< [\\_\\_stable](#), [\\_RAIter](#), [\\_StrictWeakOrdering](#) >
- struct [\\_\\_gnu\\_parallel::\\_SamplingSorter](#)< false, [\\_RAIter](#), [\\_StrictWeakOrdering](#) >

## Namespaces

- [\\_\\_gnu\\_parallel](#)

## Macros

- #define [GLIBCXX\\_PARALLEL\\_DECISION](#)([\\_\\_a](#), [\\_\\_b](#), [\\_\\_c](#), [\\_\\_d](#))
- #define [GLIBCXX\\_PARALLEL\\_LENGTH](#)([\\_\\_s](#))
- #define [GLIBCXX\\_PARALLEL\\_MERGE\\_3\\_CASE](#)([\\_\\_a](#), [\\_\\_b](#), [\\_\\_c](#), [\\_\\_c0](#), [\\_\\_c1](#))
- #define [GLIBCXX\\_PARALLEL\\_MERGE\\_4\\_CASE](#)([\\_\\_a](#), [\\_\\_b](#), [\\_\\_c](#), [\\_\\_d](#), [\\_\\_c0](#), [\\_\\_c1](#), [\\_\\_c2](#))

## Functions

- template<typename [\\_RAIter1](#) , typename [\\_RAIter2](#) , typename [\\_OutputIterator](#) , typename [\\_DifferenceTp](#) , typename [\\_Compare](#) > [\\_RAIterOutputIterator](#) [\\_\\_gnu\\_parallel::\\_merge\\_advance](#) ([\\_RAIter1](#) & [\\_\\_begin1](#), [\\_RAIter1](#) [\\_\\_end1](#), [\\_RAIter2](#) & [\\_\\_begin2](#), [\\_RAIter2](#) [\\_\\_end2](#), [\\_OutputIterator](#) [\\_\\_target](#), [\\_DifferenceTp](#) [\\_\\_max\\_length](#), [\\_Compare](#) [\\_\\_comp](#))
- template<bool [\\_\\_stable](#), bool [\\_\\_sentinels](#), typename [\\_RAIterIterator](#) , typename [\\_RAIter3](#) , typename [\\_DifferenceTp](#) , typename [\\_Compare](#) > [\\_RAIter3](#) [\\_\\_gnu\\_parallel::\\_sequential\\_multiway\\_merge](#) ([\\_RAIterIterator](#) [\\_\\_seqs\\_begin](#), [\\_RAIterIterator](#) [\\_\\_seqs\\_end](#), [\\_RAIter3](#) [\\_\\_target](#), const typename std::iterator\_traits< typename std::iterator\_traits< [\\_RAIterIterator](#) >::value\_type::first\_type >::value\_type & [\\_\\_sentinel](#), [\\_DifferenceTp](#) [\\_\\_length](#), [\\_Compare](#) [\\_\\_comp](#))
- template<typename [\\_RAIterPairIterator](#) , typename [\\_RAIterOut](#) , typename [\\_DifferenceTp](#) , typename [\\_Compare](#) > [\\_RAIterOut](#) [\\_\\_gnu\\_parallel::multiway\\_merge](#) ([\\_RAIterPairIterator](#) [\\_\\_seqs\\_begin](#), [\\_RAIterPairIterator](#) [\\_\\_seqs\\_end](#), [\\_RAIterOut](#) [\\_\\_target](#), [\\_DifferenceTp](#) [\\_\\_length](#), [\\_Compare](#) [\\_\\_comp](#), [\\_\\_gnu\\_parallel::sequential\\_tag](#))
- template<typename [\\_RAIterPairIterator](#) , typename [\\_RAIterOut](#) , typename [\\_DifferenceTp](#) , typename [\\_Compare](#) > [\\_RAIterOut](#) [\\_\\_gnu\\_parallel::multiway\\_merge](#) ([\\_RAIterPairIterator](#) [\\_\\_seqs\\_begin](#), [\\_RAIterPairIterator](#) [\\_\\_seqs\\_end](#), [\\_RAIterOut](#) [\\_\\_target](#), [\\_DifferenceTp](#) [\\_\\_length](#), [\\_Compare](#) [\\_\\_comp](#), [\\_\\_gnu\\_parallel::exact\\_tag](#) [\\_\\_tag](#))
- template<typename [\\_RAIterPairIterator](#) , typename [\\_RAIterOut](#) , typename [\\_DifferenceTp](#) , typename [\\_Compare](#) > [\\_RAIterOut](#) [\\_\\_gnu\\_parallel::multiway\\_merge](#) ([\\_RAIterPairIterator](#) [\\_\\_seqs\\_begin](#), [\\_RAIterPairIterator](#) [\\_\\_seqs\\_end](#), [\\_RAIterOut](#) [\\_\\_target](#), [\\_DifferenceTp](#) [\\_\\_length](#), [\\_Compare](#) [\\_\\_comp](#), [\\_\\_gnu\\_parallel::sampling\\_tag](#) [\\_\\_tag](#))
- template<typename [\\_RAIterPairIterator](#) , typename [\\_RAIterOut](#) , typename [\\_DifferenceTp](#) , typename [\\_Compare](#) > [\\_RAIterOut](#) [\\_\\_gnu\\_parallel::multiway\\_merge](#) ([\\_RAIterPairIterator](#) [\\_\\_seqs\\_begin](#), [\\_RAIterPairIterator](#) [\\_\\_seqs\\_end](#), [\\_RAIterOut](#) [\\_\\_target](#), [\\_DifferenceTp](#) [\\_\\_length](#), [\\_Compare](#) [\\_\\_comp](#), [parallel\\_tag](#) [\\_\\_tag](#)=[parallel\\_tag](#)(0))
- template<typename [\\_RAIterPairIterator](#) , typename [\\_RAIterOut](#) , typename [\\_DifferenceTp](#) , typename [\\_Compare](#) > [\\_RAIterOut](#) [\\_\\_gnu\\_parallel::multiway\\_merge](#) ([\\_RAIterPairIterator](#) [\\_\\_seqs\\_begin](#), [\\_RAIterPairIterator](#) [\\_\\_seqs\\_end](#), [\\_RAIterOut](#) [\\_\\_target](#), [\\_DifferenceTp](#) [\\_\\_length](#), [\\_Compare](#) [\\_\\_comp](#), [default\\_parallel\\_tag](#) [\\_\\_tag](#))
- template<template< typename [RAI](#), typename [C](#) > class iterator, typename [\\_RAIterIterator](#) , typename [\\_RAIter3](#) , typename [\\_DifferenceTp](#) , typename [\\_Compare](#) > [\\_RAIter3](#) [\\_\\_gnu\\_parallel::multiway\\_merge\\_3\\_variant](#) ([\\_RAIterIterator](#) [\\_\\_seqs\\_begin](#), [\\_RAIterIterator](#) [\\_\\_seqs\\_end](#), [\\_RAIter3](#) [\\_\\_target](#), [\\_DifferenceTp](#) [\\_\\_length](#), [\\_Compare](#) [\\_\\_comp](#))
- template<template< typename [RAI](#), typename [C](#) > class iterator, typename [\\_RAIterIterator](#) , typename [\\_RAIter3](#) , typename [\\_DifferenceTp](#) , typename [\\_Compare](#) > [\\_RAIter3](#) [\\_\\_gnu\\_parallel::multiway\\_merge\\_4\\_variant](#) ([\\_RAIterIterator](#) [\\_\\_seqs\\_begin](#), [\\_RAIterIterator](#) [\\_\\_seqs\\_end](#), [\\_RAIter3](#) [\\_\\_target](#), [\\_DifferenceTp](#) [\\_\\_length](#), [\\_Compare](#) [\\_\\_comp](#))

- `template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType> void \_\_gnu\_parallel::multiway\_merge\_exact\_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType>> *__pieces)`
- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare> _RAIter3 \_\_gnu\_parallel::multiway\_merge\_loser\_tree (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<typename UnguardedLoserTree, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare> _RAIter3 \_\_gnu\_parallel::multiway\_merge\_loser\_tree\_sentinel (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator>::value_type::first_type>::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare> _RAIter3 \_\_gnu\_parallel::multiway\_merge\_loser\_tree\_unguarded (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator>::value_type::first_type>::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType> void \_\_gnu\_parallel::multiway\_merge\_sampling\_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType>> *__pieces)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut \_\_gnu\_parallel::multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut \_\_gnu\_parallel::multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut \_\_gnu\_parallel::multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut \_\_gnu\_parallel::multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut \_\_gnu\_parallel::multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)`
- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Splitter, typename _Compare> _RAIter3 \_\_gnu\_parallel::parallel\_multiway\_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _Splitter __splitter, _DifferenceTp __length, _Compare __comp, ThreadIndex __num_threads)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut \_\_gnu\_parallel::stable\_multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut \_\_gnu\_parallel::stable\_multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut \_\_gnu\_parallel::stable\_multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut \_\_gnu\_parallel::stable\_multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut \_\_gnu\_parallel::stable\_multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)`

- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut __gnu_parallel::stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut __gnu_parallel::stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut __gnu_parallel::stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut __gnu_parallel::stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag\(0\))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut __gnu_parallel::stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`

### 5.247.1 Detailed Description

Implementation of sequential and parallel multiway merge.

Explanations on the high-speed merging routines in the appendix of

P. Sanders. Fast priority queues for cached memory. ACM Journal of Experimental Algorithmics, 5, 2000.

This file is a GNU parallel extension to the Standard C++ Library.

### 5.247.2 Macro Definition Documentation

#### 5.247.2.1 `#define GLIBCXX_PARALLEL_LENGTH( __s )`

Length of a sequence described by a pair of iterators.

Definition at line 54 of file `multiway_merge.h`.

Referenced by `__gnu_parallel::__sequential_multiway_merge()`, `__gnu_parallel::multiway_merge_exact_splitting()`, `__gnu_parallel::multiway_merge_loser_tree()`, `__gnu_parallel::multiway_merge_sampling_splitting()`, and `__gnu_parallel::parallel_multiway_merge()`.

## 5.248 `multiway_mergesort.h` File Reference

### Classes

- struct [\\_\\_gnu\\_parallel::\\_Piece< \\_DifferenceTp >](#)
- struct [\\_\\_gnu\\_parallel::\\_PMWMSSortingData< \\_RAIter >](#)
- struct [\\_\\_gnu\\_parallel::\\_SplitConsistently< \\_\\_exact, \\_RAIter, \\_Compare, \\_SortingPlacesIterator >](#)
- struct [\\_\\_gnu\\_parallel::\\_SplitConsistently< false, \\_RAIter, \\_Compare, \\_SortingPlacesIterator >](#)
- struct [\\_\\_gnu\\_parallel::\\_SplitConsistently< true, \\_RAIter, \\_Compare, \\_SortingPlacesIterator >](#)

### Namespaces

- [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _RAIter, typename _DifferenceTp> void __gnu_parallel::__determine_samples` (`_PMWMSortingData< _RAIter > * __sd, _DifferenceTp __num_samples`)
- `template<bool __stable, bool __exact, typename _RAIter, typename _Compare> void __gnu_parallel::parallel_sort_mwms` (`(__RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`)
- `template<bool __stable, bool __exact, typename _RAIter, typename _Compare> void __gnu_parallel::parallel_sort_mwms_pu` (`(_PMWMSortingData< _RAIter > * __sd, _Compare & __comp)`)

## 5.248.1 Detailed Description

Parallel multiway merge sort. This file is a GNU parallel extension to the Standard C++ Library.

5.249 `nested_exception.h` File Reference

## Classes

- class `std::nested_exception`

## Namespaces

- `std`

## Functions

- `template<typename _Ex> const nested_exception * std::__get_nested_exception` (`(const _Ex & __ex)`)
- `template<typename _Ex> void std::__throw_with_nested` (`(_Ex &&, const nested_exception * __ex) __attribute__((__noreturn__))`)
- `template<typename _Ex> void std::__throw_with_nested` (`(_Ex &&, ...) __attribute__((__noreturn__))`)
- `template<typename _Ex> void std::rethrow_if_nested` (`(const _Ex & __ex)`)
- `void std::rethrow_if_nested` (`(const nested_exception & __ex)`)
- `template<typename _Ex> void std::throw_with_nested` (`(_Ex __ex)`)

## 5.249.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

5.250 `new_allocator.h` File Reference

## Classes

- class `__gnu_cxx::new_allocator< _Tp >`

## Namespaces

- `__gnu_cxx`



## Functions

- `template<typename _Tp> bool __gnu_cxx::operator!= (const new_allocator< _Tp> &, const new_allocator< _Tp> &)`
- `template<typename _Tp> bool __gnu_cxx::operator== (const new_allocator< _Tp> &, const new_allocator< _Tp> &)`

### 5.250.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.251 node.hpp File Reference

### Classes

- [struct \\_\\_gnu\\_pbds::detail::left\\_child\\_next\\_sibling\\_heap\\_node\\_< \\_Value, \\_Metadata, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### 5.251.1 Detailed Description

Contains an implementation struct for this type of heap's node.

## 5.252 node.hpp File Reference

### Classes

- [struct \\_\\_gnu\\_pbds::detail::rb\\_tree\\_node\\_< Value\\_Type, Metadata, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### 5.252.1 Detailed Description

Contains an implementation for `rb_tree_`.

## 5.253 node.hpp File Reference

### Classes

- [struct \\_\\_gnu\\_pbds::detail::splay\\_tree\\_node\\_< Value\\_Type, Metadata, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

## 5.253.1 Detailed Description

Contains an implementation struct for splay\_tree\_'s node.

## 5.254 node\_iterators.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_const\\_node\\_it\\_< Node, Const\\_Iterator, Iterator, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_node\\_it\\_< Node, Const\\_Iterator, Iterator, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- **#define PB\_DS\_TREE\_CONST\_NODE\_ITERATOR\_CLASS\_C\_DEC**
- **#define PB\_DS\_TREE\_NODE\_ITERATOR\_CLASS\_C\_DEC**

## 5.254.1 Detailed Description

Contains an implementation class for bin\_search\_tree\_.

## 5.255 node\_iterators.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::ov\\_tree\\_node\\_const\\_it\\_< Value\\_Type, Metadata\\_Type, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::detail::ov\\_tree\\_node\\_it\\_< Value\\_Type, Metadata\\_Type, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- **#define PB\_DS\_OV\_TREE\_CONST\_NODE\_ITERATOR\_C\_DEC**
- **#define PB\_DS\_OV\_TREE\_NODE\_ITERATOR\_C\_DEC**

## 5.255.1 Detailed Description

Contains an implementation class for ov\_tree\_.

## 5.256 node\_metadata\_selector.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::tree\\_metadata\\_helper< Node\\_Update, \\_BTp >](#)
- struct [\\_\\_gnu\\_pbds::detail::tree\\_metadata\\_helper< Node\\_Update, false >](#)
- struct [\\_\\_gnu\\_pbds::detail::tree\\_metadata\\_helper< Node\\_Update, true >](#)
- struct [\\_\\_gnu\\_pbds::detail::tree\\_node\\_metadata\\_dispatch< Key, Data, Cmp\\_Fn, Node\\_Update, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### 5.256.1 Detailed Description

Contains an implementation class for trees.

## 5.257 node\_metadata\_selector.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::trie\\_metadata\\_helper< Node\\_Update, \\_BTp >](#)
- struct [\\_\\_gnu\\_pbds::detail::trie\\_metadata\\_helper< Node\\_Update, false >](#)
- struct [\\_\\_gnu\\_pbds::detail::trie\\_metadata\\_helper< Node\\_Update, true >](#)
- struct [\\_\\_gnu\\_pbds::detail::trie\\_node\\_metadata\\_dispatch< Key, Data, Cmp\\_Fn, Node\\_Update, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### 5.257.1 Detailed Description

Contains an implementation class for tries.

## 5.258 null\_node\_metadata.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::dumnode\\_const\\_iterator< Key, Data, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### 5.258.1 Detailed Description

Contains an implementation class for tree-like classes.

## 5.259 numeric\_traits.h File Reference

## Namespaces

- [\\_\\_gnu\\_cxx](#)

## Macros

- `#define __glibcxx_digits(_Tp)`
- `#define __glibcxx_digits10(_Tp)`
- `#define __glibcxx_floating(_Tp, _Fval, _Dval, _LDval)`
- `#define __glibcxx_max(_Tp)`
- `#define __glibcxx_max_digits10(_Tp)`
- `#define __glibcxx_max_exponent10(_Tp)`
- `#define __glibcxx_min(_Tp)`
- `#define __glibcxx_signed(_Tp)`

## 5.259.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.260 numeric\_fwd.h File Reference

## Namespaces

- [std](#)
- [std::\\_\\_parallel](#)

## Functions

- `template<typename _Iter, typename _Tp, typename _Tag > _Tp std::__parallel::__accumulate_switch (_Iter, _Iter, _Tp, _Tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper, typename _Tag > _Tp std::__parallel::__accumulate_switch (_Iter, _Iter, _Tp, _BinaryOper, _Tag)`
- `template<typename _RAIter, typename _Tp, typename _BinaryOper > _Tp std::__parallel::__accumulate_switch (_RAIter, _RAIter, _Tp, _BinaryOper, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism=__gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2 > _OIter std::__parallel::__adjacent_difference_switch (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper > _OIter std::__parallel::__adjacent_difference_switch (_Iter, _Iter, _OIter, _BinaryOper, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism=__gnu_parallel::parallel_unbalanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename BinaryFunction1, typename BinaryFunction2 > _Tp std::__parallel::__inner_product_switch (_RAIter1, _RAIter1, _RAIter2, _Tp, BinaryFunction1, BinaryFunction2, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::Parallelism=__gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename BinaryFunction1, typename BinaryFunction2, typename _Tag1, typename _Tag2 > _Tp std::__parallel::__inner_product_switch (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, _Tag1, _Tag2)`

- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2 > _OIter std::__parallel↵  
::__partial_sum_switch (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper > _OIter std::__parallel::__partial_sum_switch (_Iter,  
_Iter, _OIter, _BinaryOper, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp > _Tp std::__parallel::accumulate (_Iter, _Iter, _Tp)`
- `template<typename _Iter, typename _Tp > _Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, __gnu_parallel↵  
::__sequential_tag)`
- `template<typename _Iter, typename _Tp > _Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, __gnu_parallel::__↵  
Parallelism)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper > _Tp std::__parallel::accumulate (_Iter, _Iter, _Tp,  
_BinaryOper)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper > _Tp std::__parallel::accumulate (_Iter, _Iter, _Tp,  
_BinaryOper, __gnu_parallel::__sequential_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper > _Tp std::__parallel::accumulate (_Iter, _Iter, _Tp,  
_BinaryOper, __gnu_parallel::__Parallelism)`
- `template<typename _Iter, typename _OIter > _OIter std::__parallel::adjacent_difference (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper > _OIter std::__parallel::adjacent_difference (_Iter,  
_Iter, _OIter, _BinaryOper)`
- `template<typename _Iter, typename _OIter > _OIter std::__parallel::adjacent_difference (_Iter, _Iter, _OIter, __↵  
gnu_parallel::__sequential_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper > _OIter std::__parallel::adjacent_difference (_Iter,  
_Iter, _OIter, _BinaryOper, __gnu_parallel::__sequential_tag)`
- `template<typename _Iter, typename _OIter > _OIter std::__parallel::adjacent_difference (_Iter, _Iter, _OIter, __↵  
gnu_parallel::__Parallelism)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper > _OIter std::__parallel::adjacent_difference (_Iter,  
_Iter, _OIter, _BinaryOper, __gnu_parallel::__Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp > _Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2,  
_Tp)`
- `template<typename _Iter1, typename _Iter2, typename _Tp > _Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2,  
_Tp, __gnu_parallel::__sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp > _Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2,  
_Tp, __gnu_parallel::__Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 > _Tp std↵  
::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 > _Tp std::__↵  
__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, __gnu_parallel↵  
::__sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename BinaryFunction1, typename BinaryFunction2 > _Tp std::__↵  
__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, BinaryFunction1, BinaryFunction2, __gnu_parallel::__↵  
Parallelism)`
- `template<typename _Iter, typename _OIter > _OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter, __gnu↵  
parallel::__sequential_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper > _OIter std::__parallel::partial_sum (_Iter, _Iter,  
_OIter, _BinaryOper, __gnu_parallel::__sequential_tag)`
- `template<typename _Iter, typename _OIter > _OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter __result)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper > _OIter std::__parallel::partial_sum (_Iter, _Iter,  
_OIter, _BinaryOper)`

### 5.260.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

## 5.261 omp\_loop.h File Reference

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result > _Op __gnu_parallel::__for↔  
_each_template_random_access_omp_loop (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r,  
_Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::difference_type __bound)`

#### 5.261.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop. This file is a GNU parallel extension to the Standard C++ Library.

## 5.262 omp\_loop\_static.h File Reference

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result > _Op __gnu_parallel::__for↔  
_each_template_random_access_omp_loop_static (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red  
__r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::difference_type __bound)`

#### 5.262.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop with static scheduling. This file is a GNU parallel extension to the Standard C++ Library.

## 5.263 opt\_random.h File Reference

### Namespaces

- [std](#)

#### 5.263.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<random>`.

## 5.264 `order_statistics_imp.hpp` File Reference

### 5.264.1 Detailed Description

Contains forward declarations for `order_statistics_key`

## 5.265 `order_statistics_imp.hpp` File Reference

### 5.265.1 Detailed Description

Contains forward declarations for `order_statistics_key`

## 5.266 `os_defines.h` File Reference

### Macros

- `#define __NO_CTYPE`

### 5.266.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

## 5.267 `ostream_insert.h` File Reference

### Namespaces

- [std](#)

### Functions

- `template<typename _CharT, typename _Traits> void std::__ostream_fill (basic_ostream< _CharT, _Traits> &__out, streamsize __n)`
- `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits> & std::__ostream_insert (basic_ostream< _CharT, _Traits> &__out, const _CharT *__s, streamsize __n)`
- `template<typename _CharT, typename _Traits> void std::__ostream_write (basic_ostream< _CharT, _Traits> &__out, const _CharT *__s, streamsize __n)`

### 5.267.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ostream>`.

## 5.268 `ov_tree_map.hpp` File Reference

### Classes

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

- class [\\_\\_gnu\\_pbds::detail::ov\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >::cond\_dtor< Size\_Type >

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### Macros

- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_CONST\_NODE\_ITERATOR\_NAME**
- #define **PB\_DS\_OV\_TREE\_NAME**
- #define **PB\_DS\_OV\_TREE\_TRAITS\_BASE**

#### 5.268.1 Detailed Description

Contains an implementation class for ov\_tree.

## 5.269 pairing\_heap\_.hpp File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::detail::pairing\\_heap](#)< Value\_Type, Cmp\_Fn, \_Alloc >

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### Macros

- #define **PB\_DS\_ASSERT\_NODE\_CONSISTENT**(\_Node, \_Bool)
- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_P\_HEAP\_BASE**

#### 5.269.1 Detailed Description

Contains an implementation class for a pairing heap.

## 5.270 par\_loop.h File Reference

#### Namespaces

- [\\_\\_gnu\\_parallel](#)



## Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result > _Op \_\_gnu\_parallel::\_\_for↔  
\_\_each\_template\_random\_access\_ed (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result  
__base, _Result &__output, typename std::iterator_traits< _RAIter >::difference_type __bound)`

### 5.270.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of equal splitting. This file is a GNU parallel extension to the Standard C++ Library.

## 5.271 `parallel.h` File Reference

### 5.271.1 Detailed Description

End-user include file. Provides advanced settings and tuning options. This file is a GNU parallel extension to the Standard C++ Library.

## 5.272 `partial_sum.h` File Reference

## Namespaces

- [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation > _OutputIterator \_\_gnu\_parallel::\_\_↔  
parallel\_partial\_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation > _OutputIterator \_\_gnu\_parallel::\_\_↔  
parallel\_partial\_sum\_basecase (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __↔  
bin_op, typename std::iterator_traits< _Iter >::value_type __value)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation > _OutputIterator \_\_gnu\_parallel::\_\_↔  
parallel\_partial\_sum\_linear (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op,  
typename std::iterator_traits< _Iter >::difference_type __n)`

### 5.272.1 Detailed Description

Parallel implementation of `std::partial_sum()`, i.e. prefix sums. This file is a GNU parallel extension to the Standard C++ Library.

## 5.273 `partition.h` File Reference

## Namespaces

- [\\_\\_gnu\\_parallel](#)

## Macros

- `#define \_GLIBCXX\_VOLATILE`

## Functions

- `template<typename _RAIter, typename _Compare> void \_\_gnu\_parallel::\_\_parallel\_nth\_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare> void \_\_gnu\_parallel::\_\_parallel\_partial\_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Predicate> std::iterator_traits< _RAIter >::difference_type \_\_gnu\_parallel::\_\_parallel\_partition (_RAIter __begin, _RAIter __end, _Predicate __pred, _ThreadIndex __num_threads)`

## 5.273.1 Detailed Description

Parallel implementation of `std::partition()`, `std::nth_element()`, and `std::partial_sort()`. This file is a GNU parallel extension to the Standard C++ Library.

## 5.273.2 Macro Definition Documentation

5.273.2.1 `#define GLIBCXX_VOLATILE`

Decide whether to declare certain variables volatile.

Definition at line 43 of file `partition.h`.

Referenced by `__gnu_parallel::__parallel_partition()`.

## 5.274 pat\_trie.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_map< Key, Mapped, Node\\_And\\_It\\_Traits, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_ASSERT_NODE_VALID(X)`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_PAT_TRIE_NAME`
- `#define PB_DS_PAT_TRIE_TRAITS_BASE`
- `#define PB_DS_RECURSIVE_COUNT_LEAFS(X)`

## 5.274.1 Detailed Description

Contains an implementation class for a patricia tree.

## 5.275 pat\_trie\_base.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base](#)
- class [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Clter< Node, Leaf, Head, Inode, Is\\_Forward\\_Iterator >](#)
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Head< \\_ATraits, Metadata >](#)
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Inode< \\_ATraits, Metadata >](#)
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Inode< \\_ATraits, Metadata >::const\\_iterator](#)
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Inode< \\_ATraits, Metadata >::iterator](#)
- class [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Iter< Node, Leaf, Head, Inode, Is\\_Forward\\_Iterator >](#)
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Leaf< \\_ATraits, Metadata >](#)
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Metadata< Metadata, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Metadata< null\\_type, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_base< \\_ATraits, Metadata >](#)
- class [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_citer< Node, Leaf, Head, Inode, \\_Clterator, Iterator, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_iter< Node, Leaf, Head, Inode, \\_Clterator, Iterator, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CONST_IT_C_DEC`
- `#define PB_DS_CONST_ODIR_IT_C_DEC`
- `#define PB_DS_IT_C_DEC`
- `#define PB_DS_ODIR_IT_C_DEC`
- `#define PB_DS_PAT_TRIE_NODE_CONST_ITERATOR_C_DEC`
- `#define PB_DS_PAT_TRIE_NODE_ITERATOR_C_DEC`

#### 5.275.1 Detailed Description

Contains the base class for a patricia tree.

## 5.276 pod\_char\_traits.h File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::character< V, I, S >](#)
- struct [std::char\\_traits< \\_\\_gnu\\_cxx::character< V, I, S > >](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

## Functions

- `template<typename V , typename I , typename S > bool __gnu_cxx::operator< (const character< V, I, S > &lhs, const character< V, I, S > &rhs)`
- `template<typename V , typename I , typename S > bool __gnu_cxx::operator== (const character< V, I, S > &lhs, const character< V, I, S > &rhs)`

### 5.276.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.277 point\_const\_iterator.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::binary\\_heap\\_point\\_const\\_iterator\\_< Value\\_Type, Entry, Simple, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### 5.277.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

## 5.278 point\_const\_iterator.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::left\\_child\\_next\\_sibling\\_heap\\_node\\_point\\_const\\_iterator\\_< Node, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

### 5.278.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

## 5.279 `point_const_iterator.hpp` File Reference

### Classes

- class [point\\_const\\_iterator\\_](#)

### 5.279.1 Detailed Description

Contains an iterator class returned by the tables' const find and insert methods.

## 5.280 `point_iterator.hpp` File Reference

### Classes

- class [point\\_iterator\\_](#)

### 5.280.1 Detailed Description

Contains an iterator class returned by the tables' find and insert methods.

## 5.281 `point_iterators.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_TREE_CONST_IT_C_DEC`
- `#define PB_DS_TREE_CONST_ODIR_IT_C_DEC`
- `#define PB_DS_TREE_IT_C_DEC`
- `#define PB_DS_TREE_ODIR_IT_C_DEC`

### 5.281.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

## 5.282 pointer.h File Reference

## Classes

- struct [\\_\\_gnu\\_cxx::Invalid\\_type](#)
- class [\\_\\_gnu\\_cxx::Pointer\\_adapter<\\_Storage\\_policy>](#)
- class [\\_\\_gnu\\_cxx::Relative\\_pointer\\_impl<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::Relative\\_pointer\\_impl<const \\_Tp>](#)
- class [\\_\\_gnu\\_cxx::Std\\_pointer\\_impl<\\_Tp>](#)
- struct [\\_\\_gnu\\_cxx::Unqualified\\_type<\\_Tp>](#)

## Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

## Macros

- `#define CXX_POINTER_ARITH_OPERATOR_SET(INT_TYPE)`
- `#define GCC_CXX_POINTER_COMPARISON_OPERATION_SET(OPERATOR)`

## Functions

- `template<typename _Tp1, typename _Tp2> bool __gnu_cxx::operator!=(const _Pointer_adapter<_Tp1> &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2> bool __gnu_cxx::operator!=(_Tp1 __lhs, const _Pointer_adapter<_Tp2> &__rhs)`
- `template<typename _Tp1, typename _Tp2> bool __gnu_cxx::operator!=(const _Pointer_adapter<_Tp1> &__lhs, const _Pointer_adapter<_Tp2> &__rhs)`
- `template<typename _Tp> bool __gnu_cxx::operator!=(const _Pointer_adapter<_Tp> &__lhs, int __rhs)`
- `template<typename _Tp> bool __gnu_cxx::operator!=(int __lhs, const _Pointer_adapter<_Tp> &__rhs)`
- `template<typename _Tp> bool __gnu_cxx::operator!=(const _Pointer_adapter<_Tp> &__lhs, const _Pointer_adapter<_Tp> &__rhs)`
- `template<typename _Tp1, typename _Tp2> bool __gnu_cxx::operator<(_Tp1 __lhs, const _Pointer_adapter<_Tp2> &__rhs)`
- `template<typename _Tp1, typename _Tp2> bool __gnu_cxx::operator<(const _Pointer_adapter<_Tp1> &__lhs, const _Pointer_adapter<_Tp2> &__rhs)`
- `template<typename _Tp1, typename _Tp2> bool __gnu_cxx::operator<(const _Pointer_adapter<_Tp1> &__lhs, _Tp2 __rhs)`
- `template<typename _CharT, typename _Traits, typename _StoreT> std::basic_ostream<_CharT, _Traits> & __gnu_cxx::operator<<(std::basic_ostream<_CharT, _Traits> &__os, const _Pointer_adapter<_StoreT> &__p)`
- `template<typename _Tp1, typename _Tp2> bool __gnu_cxx::operator<=(const _Pointer_adapter<_Tp1> &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2> bool __gnu_cxx::operator<=(_Tp1 __lhs, const _Pointer_adapter<_Tp2> &__rhs)`
- `template<typename _Tp1, typename _Tp2> bool __gnu_cxx::operator<=(const _Pointer_adapter<_Tp1> &__lhs, const _Pointer_adapter<_Tp2> &__rhs)`
- `template<typename _Tp> bool __gnu_cxx::operator<=(const _Pointer_adapter<_Tp> &__lhs, const _Pointer_adapter<_Tp> &__rhs)`
- `template<typename _Tp1, typename _Tp2> bool __gnu_cxx::operator==(_Tp1 __lhs, const _Pointer_adapter<_Tp2> &__rhs)`

- `template<typename _Tp1, typename _Tp2> bool __gnu_cxx::operator==(const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2> bool __gnu_cxx::operator==(const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp> bool __gnu_cxx::operator==(const _Pointer_adapter< _Tp > &__lhs, int __rhs)`
- `template<typename _Tp> bool __gnu_cxx::operator==(int __lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp> bool __gnu_cxx::operator==(const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2> bool __gnu_cxx::operator>(_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2> bool __gnu_cxx::operator>(const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2> bool __gnu_cxx::operator>(const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp> bool __gnu_cxx::operator>(const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2> bool __gnu_cxx::operator>=(const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2> bool __gnu_cxx::operator>=(const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2> bool __gnu_cxx::operator>=(_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp> bool __gnu_cxx::operator>=(const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`

### 5.282.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

#### Author

Bob Walters

Provides reusable `_Pointer_adapter` for assisting in the development of custom pointer types that can be used with the standard containers via the `allocator::pointer` and `allocator::const_pointer` typedefs.

## 5.283 `policy_access_fn_imps.hpp` File Reference

### 5.283.1 Detailed Description

Contains an implementation class for a `binary_heap`.

## 5.284 `policy_access_fn_imps.hpp` File Reference

### 5.284.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

## 5.285 policy\_access\_fn\_imps.hpp File Reference

### 5.285.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s policy access functions.

## 5.286 policy\_access\_fn\_imps.hpp File Reference

### 5.286.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s policy access functions.

## 5.287 policy\_access\_fn\_imps.hpp File Reference

### 5.287.1 Detailed Description

Contains an implementation class for left\_child\_next\_sibling\_heap\_.

## 5.288 policy\_access\_fn\_imps.hpp File Reference

### 5.288.1 Detailed Description

Contains an implementation class for ov\_tree.

## 5.289 policy\_access\_fn\_imps.hpp File Reference

### 5.289.1 Detailed Description

Contains an implementation class for pat\_trie.

## 5.290 pool\_allocator.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_pool\\_alloc<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::\\_\\_pool\\_alloc\\_base](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### Functions

- template<typename \_Tp> bool [\\_\\_gnu\\_cxx::operator!=](#) (const \_\_pool\_alloc<\_Tp> &, const \_\_pool\_alloc<\_Tp> &)
- template<typename \_Tp> bool [\\_\\_gnu\\_cxx::operator==](#) (const \_\_pool\_alloc<\_Tp> &, const \_\_pool\_alloc<\_Tp> &)



### 5.290.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.291 postypes.h File Reference

### Classes

- class [std::fpos<\\_StateT>](#)

### Namespaces

- [std](#)

### Typedefs

- typedef long long [std::streamoff](#)
- typedef fpos< mbstate\_t > [std::streampos](#)
- typedef ptrdiff\_t [std::streamsize](#)
- typedef fpos< mbstate\_t > [std::u16streampos](#)
- typedef fpos< mbstate\_t > [std::u32streampos](#)
- typedef fpos< mbstate\_t > [std::wstreampos](#)

### Functions

- template<typename \_StateT> bool **std::operator!=** (const fpos< \_StateT > &\_\_lhs, const fpos< \_StateT > &\_\_rhs)
- template<typename \_StateT> bool [std::operator==](#) (const fpos< \_StateT > &\_\_lhs, const fpos< \_StateT > &\_\_rhs)

### 5.291.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

## 5.292 prefix\_search\_node\_update\_imp.hpp File Reference

### 5.292.1 Detailed Description

Contains an implementation of `prefix_search_node_update`.

## 5.293 priority\_queue.hpp File Reference

### Classes

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

### Namespaces

- [\\_\\_gnu\\_pbds](#)

## 5.293.1 Detailed Description

Contains `priority_queues`.

5.294 `priority_queue_base_dispatch.hpp` File Reference

## Classes

- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch<\\_VTp, Cmp\\_Fn, \\_Alloc, binary\\_heap\\_tag, null\\_type>](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch<\\_VTp, Cmp\\_Fn, \\_Alloc, binomial\\_heap\\_tag, null\\_type>](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch<\\_VTp, Cmp\\_Fn, \\_Alloc, pairing\\_heap\\_tag, null\\_type>](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch<\\_VTp, Cmp\\_Fn, \\_Alloc, rc\\_binomial\\_heap\\_tag, null\\_type>](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch<\\_VTp, Cmp\\_Fn, \\_Alloc, thin\\_heap\\_tag, null\\_type>](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_ASSERT_VALID(X)`
- `#define PB_DS_DEBUG_VERIFY(_Cond)`

## 5.294.1 Detailed Description

Contains an pqiative container dispatching base.

5.295 `probe_fn_base.hpp` File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::probe\\_fn\\_base<\\_Alloc>](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## 5.295.1 Detailed Description

Contains a probe policy base.

5.296 `profiler.h` File Reference

## Classes

- struct [\\_\\_gnu\\_profile::\\_\\_reentrance\\_guard](#)

## Namespaces

- [\\_\\_gnu\\_profile](#)

## Macros

- #define **\_\_profcxx\_hashtable\_construct**(\_\_x...)
- #define **\_\_profcxx\_hashtable\_construct2**(\_\_x...)
- #define **\_\_profcxx\_hashtable\_destruct**(\_\_x...)
- #define **\_\_profcxx\_hashtable\_destruct2**(\_\_x...)
- #define **\_\_profcxx\_hashtable\_resize**(\_\_x...)
- #define **\_\_profcxx\_inefficient\_hash\_is\_on**()
- #define **\_\_profcxx\_is\_invalid**()
- #define **\_\_profcxx\_is\_off**()
- #define **\_\_profcxx\_is\_on**()
- #define **\_\_profcxx\_list\_construct**(\_\_x...)
- #define **\_\_profcxx\_list\_construct2**(\_\_x...)
- #define **\_\_profcxx\_list\_destruct**(\_\_x...)
- #define **\_\_profcxx\_list\_destruct2**(\_\_x...)
- #define **\_\_profcxx\_list\_insert**(\_\_x...)
- #define **\_\_profcxx\_list\_invalid\_operator**(\_\_x...)
- #define **\_\_profcxx\_list\_iterate**(\_\_x...)
- #define **\_\_profcxx\_list\_operation**(\_\_x...)
- #define **\_\_profcxx\_list\_rewind**(\_\_x...)
- #define **\_\_profcxx\_map\_to\_unordered\_map\_construct**(\_\_x...)
- #define **\_\_profcxx\_map\_to\_unordered\_map\_destruct**(\_\_x...)
- #define **\_\_profcxx\_map\_to\_unordered\_map\_erase**(\_\_x...)
- #define **\_\_profcxx\_map\_to\_unordered\_map\_find**(\_\_x...)
- #define **\_\_profcxx\_map\_to\_unordered\_map\_insert**(\_\_x...)
- #define **\_\_profcxx\_map\_to\_unordered\_map\_invalidate**(\_\_x...)
- #define **\_\_profcxx\_map\_to\_unordered\_map\_iterate**(\_\_x...)
- #define **\_\_profcxx\_report**()
- #define **\_\_profcxx\_turn\_off**()
- #define **\_\_profcxx\_turn\_on**()
- #define **\_\_profcxx\_vector\_construct**(\_\_x...)
- #define **\_\_profcxx\_vector\_construct2**(\_\_x...)
- #define **\_\_profcxx\_vector\_destruct**(\_\_x...)
- #define **\_\_profcxx\_vector\_destruct2**(\_\_x...)
- #define **\_\_profcxx\_vector\_find**(\_\_x...)
- #define **\_\_profcxx\_vector\_insert**(\_\_x...)
- #define **\_\_profcxx\_vector\_invalid\_operator**(\_\_x...)
- #define **\_\_profcxx\_vector\_iterate**(\_\_x...)
- #define **\_\_profcxx\_vector\_resize**(\_\_x...)
- #define **\_\_profcxx\_vector\_resize2**(\_\_x...)
- #define **GLIBCXX\_PROFILE\_DATA**(\_\_name)
- #define **GLIBCXX\_PROFILE\_DEFINE\_DATA**(\_\_type, \_\_name, \_\_initial\_value...)
- #define **GLIBCXX\_PROFILE\_DEFINE\_UNINIT\_DATA**(\_\_type, \_\_name)
- #define **GLIBCXX\_PROFILE\_MAX\_STACK\_DEPTH**
- #define **GLIBCXX\_PROFILE\_MAX\_STACK\_DEPTH\_ENV\_VAR**
- #define **GLIBCXX\_PROFILE\_MAX\_WARN\_COUNT**

- `#define _GLIBCXX_PROFILE_MAX_WARN_COUNT_ENV_VAR`
- `#define _GLIBCXX_PROFILE_MEM_PER_DIAGNOSTIC`
- `#define _GLIBCXX_PROFILE_MEM_PER_DIAGNOSTIC_ENV_VAR`
- `#define _GLIBCXX_PROFILE_REENTRANCE_GUARD(__x...)`
- `#define _GLIBCXX_PROFILE_TRACE_ENV_VAR`
- `#define _GLIBCXX_PROFILE_TRACE_PATH_ROOT`

## Functions

- `bool __gnu_profile::is_invalid ()`
- `bool __gnu_profile::is_off ()`
- `bool __gnu_profile::is_on ()`
- `void __gnu_profile::report (void)`
- `void __gnu_profile::trace_hash_func_construct (const void *)`
- `void __gnu_profile::trace_hash_func_destruct (const void *, std::size_t, std::size_t, std::size_t)`
- `void __gnu_profile::trace_hashtable_size_construct (const void *, std::size_t)`
- `void __gnu_profile::trace_hashtable_size_destruct (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::trace_hashtable_size_resize (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::trace_list_to_set_construct (const void *)`
- `void __gnu_profile::trace_list_to_set_destruct (const void *)`
- `void __gnu_profile::trace_list_to_set_find (const void *, std::size_t)`
- `void __gnu_profile::trace_list_to_set_insert (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::trace_list_to_set_invalid_operator (const void *)`
- `void __gnu_profile::trace_list_to_set_iterate (const void *, std::size_t)`
- `void __gnu_profile::trace_list_to_slist_construct (const void *)`
- `void __gnu_profile::trace_list_to_slist_destruct (const void *)`
- `void __gnu_profile::trace_list_to_slist_operation (const void *)`
- `void __gnu_profile::trace_list_to_slist_rewind (const void *)`
- `void __gnu_profile::trace_list_to_vector_construct (const void *)`
- `void __gnu_profile::trace_list_to_vector_destruct (const void *)`
- `void __gnu_profile::trace_list_to_vector_insert (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::trace_list_to_vector_invalid_operator (const void *)`
- `void __gnu_profile::trace_list_to_vector_iterate (const void *, std::size_t)`
- `void __gnu_profile::trace_list_to_vector_resize (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::trace_map_to_unordered_map_construct (const void *)`
- `void __gnu_profile::trace_map_to_unordered_map_destruct (const void *)`
- `void __gnu_profile::trace_map_to_unordered_map_erase (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::trace_map_to_unordered_map_find (const void *, std::size_t)`
- `void __gnu_profile::trace_map_to_unordered_map_insert (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::trace_map_to_unordered_map_invalidate (const void *)`
- `void __gnu_profile::trace_map_to_unordered_map_iterate (const void *, std::size_t)`
- `void __gnu_profile::trace_vector_size_construct (const void *, std::size_t)`
- `void __gnu_profile::trace_vector_size_destruct (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::trace_vector_size_resize (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::trace_vector_to_list_construct (const void *)`
- `void __gnu_profile::trace_vector_to_list_destruct (const void *)`
- `void __gnu_profile::trace_vector_to_list_find (const void *, std::size_t)`
- `void __gnu_profile::trace_vector_to_list_insert (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::trace_vector_to_list_invalid_operator (const void *)`
- `void __gnu_profile::trace_vector_to_list_iterate (const void *, std::size_t)`
- `void __gnu_profile::trace_vector_to_list_resize (const void *, std::size_t, std::size_t)`
- `bool __gnu_profile::turn_off ()`
- `bool __gnu_profile::turn_on ()`

### 5.296.1 Detailed Description

Interface of the profiling runtime library.

## 5.297 profiler\_algos.h File Reference

### Namespaces

- [\\_\\_gnu\\_profile](#)

### Functions

- `template<typename _InputIterator, typename _Function> _Function __gnu_profile::__for_each (_InputIterator __first, ↵  
_InputIterator __last, _Function __f)`
- `template<typename _Container> void __gnu_profile::__insert_top_n (_Container &__output, const typename _↵  
_Container::value_type &__value, typename _Container::size_type __n)`
- `template<typename _ForwardIterator, typename _Tp> _ForwardIterator __gnu_profile::__remove (_ForwardIterator _↵  
__first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _Container> void __gnu_profile::__top_n (const _Container &__input, _Container &__output,  
typename _Container::size_type __n)`

### 5.297.1 Detailed Description

Algorithms used by the profile extension.

This file is needed to avoid including `<algorithm>` or `<bits/stl_algo.h>`. Including those files would result in recursive includes. These implementations are oversimplified. In general, efficiency may be sacrificed to minimize maintenance overhead.

## 5.298 profiler\_container\_size.h File Reference

### Classes

- class [\\_\\_gnu\\_profile::\\_\\_container\\_size\\_info](#)
- class [\\_\\_gnu\\_profile::\\_\\_container\\_size\\_stack\\_info](#)
- class [\\_\\_gnu\\_profile::\\_\\_trace\\_container\\_size](#)

### Namespaces

- [\\_\\_gnu\\_profile](#)

### 5.298.1 Detailed Description

Diagnostics for container sizes.

## 5.299 profiler\_hash\_func.h File Reference

### Classes

- class [\\_\\_gnu\\_profile::\\_\\_hashfunc\\_info](#)
- class [\\_\\_gnu\\_profile::\\_\\_hashfunc\\_stack\\_info](#)
- class [\\_\\_gnu\\_profile::\\_\\_trace\\_hash\\_func](#)

### Namespaces

- [\\_\\_gnu\\_profile](#)

### Functions

- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hash\\_func\\_construct](#) (const void \*)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hash\\_func\\_destruct](#) (const void \*, std::size\_t, std::size\_t, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hash\\_func\\_init](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hash\\_func\\_report](#) (FILE \*\_\_f, \_\_warning\_vector\_t &\_\_warnings)

#### 5.299.1 Detailed Description

Data structures to represent profiling traces.

## 5.300 profiler\_hashtable\_size.h File Reference

### Classes

- class [\\_\\_gnu\\_profile::\\_\\_trace\\_hashtable\\_size](#)

### Namespaces

- [\\_\\_gnu\\_profile](#)

### Functions

- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hashtable\\_size\\_construct](#) (const void \*, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hashtable\\_size\\_destruct](#) (const void \*, std::size\_t, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hashtable\\_size\\_init](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hashtable\\_size\\_report](#) (FILE \*\_\_f, \_\_warning\_vector\_t &\_\_warnings)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hashtable\\_size\\_resize](#) (const void \*, std::size\_t, std::size\_t)

#### 5.300.1 Detailed Description

Collection of hashtable size traces.

## 5.301 profiler\_list\_to\_slist.h File Reference

### Namespaces

- [\\_\\_gnu\\_profile](#)

### Functions

- void **\_\_gnu\_profile::\_\_trace\_list\_to\_slist\_construct** (const void \*)
- void **\_\_gnu\_profile::\_\_trace\_list\_to\_slist\_destruct** (const void \*)
- void **\_\_gnu\_profile::\_\_trace\_list\_to\_slist\_init** ()
- void **\_\_gnu\_profile::\_\_trace\_list\_to\_slist\_operation** (const void \*)
- void **\_\_gnu\_profile::\_\_trace\_list\_to\_slist\_report** (FILE \*\_\_f, \_\_warning\_vector\_t &\_\_warnings)
- void **\_\_gnu\_profile::\_\_trace\_list\_to\_slist\_rewind** (const void \*)

### 5.301.1 Detailed Description

Diagnostics for list to slist.

## 5.302 profiler\_list\_to\_vector.h File Reference

### Classes

- class [\\_\\_gnu\\_profile::\\_\\_list2vector\\_info](#)

### Namespaces

- [\\_\\_gnu\\_profile](#)

### Functions

- void **\_\_gnu\_profile::\_\_trace\_list\_to\_vector\_construct** (const void \*)
- void **\_\_gnu\_profile::\_\_trace\_list\_to\_vector\_destruct** (const void \*)
- void **\_\_gnu\_profile::\_\_trace\_list\_to\_vector\_init** ()
- void **\_\_gnu\_profile::\_\_trace\_list\_to\_vector\_insert** (const void \*, std::size\_t, std::size\_t)
- void **\_\_gnu\_profile::\_\_trace\_list\_to\_vector\_invalid\_operator** (const void \*)
- void **\_\_gnu\_profile::\_\_trace\_list\_to\_vector\_iterate** (const void \*, std::size\_t)
- void **\_\_gnu\_profile::\_\_trace\_list\_to\_vector\_report** (FILE \*\_\_f, \_\_warning\_vector\_t &\_\_warnings)
- void **\_\_gnu\_profile::\_\_trace\_list\_to\_vector\_resize** (const void \*, std::size\_t, std::size\_t)

### 5.302.1 Detailed Description

diagnostics for list to vector.

## 5.303 profiler\_map\_to\_unordered\_map.h File Reference

### Classes

- class [\\_\\_gnu\\_profile::\\_\\_map2umap\\_info](#)
- class [\\_\\_gnu\\_profile::\\_\\_map2umap\\_stack\\_info](#)
- class [\\_\\_gnu\\_profile::\\_\\_trace\\_map2umap](#)

### Namespaces

- [\\_\\_gnu\\_profile](#)

### Functions

- int [\\_\\_gnu\\_profile::\\_\\_log2](#) (std::size\_t \_\_size)
- float [\\_\\_gnu\\_profile::\\_\\_map\\_erase\\_cost](#) (std::size\_t \_\_size)
- float [\\_\\_gnu\\_profile::\\_\\_map\\_find\\_cost](#) (std::size\_t \_\_size)
- float [\\_\\_gnu\\_profile::\\_\\_map\\_insert\\_cost](#) (std::size\_t \_\_size)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_construct](#) (const void \*)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_destruct](#) (const void \*)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_erase](#) (const void \*, std::size\_t, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_find](#) (const void \*, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_init](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_insert](#) (const void \*, std::size\_t, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_invalidate](#) (const void \*)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_iterate](#) (const void \*, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_report](#) (FILE \*\_\_f, \_\_warning\_vector\_t & \_\_warnings)

### 5.303.1 Detailed Description

Diagnostics for map to unordered\_map.

## 5.304 profiler\_node.h File Reference

### Classes

- class [\\_\\_gnu\\_profile::\\_\\_object\\_info\\_base](#)
- class [\\_\\_gnu\\_profile::\\_\\_stack\\_hash](#)
- class [\\_\\_gnu\\_profile::\\_\\_stack\\_info\\_base](#)< [\\_\\_object\\_info](#) >

### Namespaces

- [\\_\\_gnu\\_profile](#)

### Typedefs

- typedef void \* [\\_\\_gnu\\_profile::\\_\\_instruction\\_address\\_t](#)
- typedef const void \* [\\_\\_gnu\\_profile::\\_\\_object\\_t](#)
- typedef std::vector< [\\_\\_instruction\\_address\\_t](#) > [\\_\\_gnu\\_profile::\\_\\_stack\\_npt](#)
- typedef [\\_\\_stack\\_npt](#) \* [\\_\\_gnu\\_profile::\\_\\_stack\\_t](#)



## Functions

- `__stack_t __gnu_profile::__get_stack ()`
- `std::size_t __gnu_profile::__size (__stack_t __stack)`
- `std::size_t __gnu_profile::__stack_max_depth ()`
- `void __gnu_profile::__write (FILE *__f, __stack_t __stack)`

### 5.304.1 Detailed Description

Data structures to represent a single profiling event.

## 5.305 profiler\_state.h File Reference

### Namespaces

- [\\_\\_gnu\\_profile](#)

### Enumerations

- `enum __state_type { __ON, __OFF, __INVALID }`

## Functions

- `bool __gnu_profile::__is_invalid ()`
- `bool __gnu_profile::__is_off ()`
- `bool __gnu_profile::__is_on ()`
- `bool __gnu_profile::__turn (__state_type __s)`
- `bool __gnu_profile::__turn_off ()`
- `bool __gnu_profile::__turn_on ()`
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA (__state_type, __state, __INVALID)`

### 5.305.1 Detailed Description

Global profiler state.

## 5.306 profiler\_trace.h File Reference

### Classes

- `class __gnu_profile::__trace_base< __object_info, __stack_info >`
- `struct __gnu_profile::__warning_data`

### Namespaces

- [\\_\\_gnu\\_profile](#)

## Macros

- `#define _GLIBCXX_IMPL_UNORDERED_MAP`

## Typedefs

- typedef std::vector< \_\_cost\_factor \* > **\_\_gnu\_profile::\_\_cost\_factor\_vector**
- typedef std::unordered\_map< std::string, std::string > **\_\_gnu\_profile::\_\_env\_t**
- typedef std::vector< \_\_warning\_data > **\_\_gnu\_profile::\_\_warning\_vector\_t**

## Functions

- std::size\_t **\_\_gnu\_profile::\_\_env\_size\_t** (const char \* \_\_env\_var, std::size\_t \_\_default\_value)
- int **\_\_gnu\_profile::\_\_log\_magnitude** (float \_\_f)
- std::size\_t **\_\_gnu\_profile::\_\_max\_mem** ()
- FILE \* **\_\_gnu\_profile::\_\_open\_output\_file** (const char \* \_\_extension)
- bool **\_\_gnu\_profile::\_\_profcxx\_init** ()
- void **\_\_gnu\_profile::\_\_profcxx\_init\_unconditional** ()
- void **\_\_gnu\_profile::\_\_read\_cost\_factors** ()
- void **\_\_gnu\_profile::\_\_report** (void)
- void **\_\_gnu\_profile::\_\_set\_cost\_factors** ()
- void **\_\_gnu\_profile::\_\_set\_max\_mem** ()
- void **\_\_gnu\_profile::\_\_set\_max\_stack\_trace\_depth** ()
- void **\_\_gnu\_profile::\_\_set\_max\_warn\_count** ()
- void **\_\_gnu\_profile::\_\_set\_trace\_path** ()
- std::size\_t **\_\_gnu\_profile::\_\_stack\_max\_depth** ()
- void **\_\_gnu\_profile::\_\_trace\_hash\_func\_init** ()
- void **\_\_gnu\_profile::\_\_trace\_hash\_func\_report** (FILE \* \_\_f, \_\_warning\_vector\_t & \_\_warnings)
- void **\_\_gnu\_profile::\_\_trace\_hashtable\_size\_init** ()
- void **\_\_gnu\_profile::\_\_trace\_hashtable\_size\_report** (FILE \* \_\_f, \_\_warning\_vector\_t & \_\_warnings)
- void **\_\_gnu\_profile::\_\_trace\_list\_to\_slist\_init** ()
- void **\_\_gnu\_profile::\_\_trace\_list\_to\_slist\_report** (FILE \* \_\_f, \_\_warning\_vector\_t & \_\_warnings)
- void **\_\_gnu\_profile::\_\_trace\_list\_to\_vector\_init** ()
- void **\_\_gnu\_profile::\_\_trace\_list\_to\_vector\_report** (FILE \* \_\_f, \_\_warning\_vector\_t & \_\_warnings)
- void **\_\_gnu\_profile::\_\_trace\_map\_to\_unordered\_map\_init** ()
- void **\_\_gnu\_profile::\_\_trace\_map\_to\_unordered\_map\_report** (FILE \* \_\_f, \_\_warning\_vector\_t & \_\_warnings)
- void **\_\_gnu\_profile::\_\_trace\_vector\_size\_init** ()
- void **\_\_gnu\_profile::\_\_trace\_vector\_size\_report** (FILE \*, \_\_warning\_vector\_t &)
- void **\_\_gnu\_profile::\_\_trace\_vector\_to\_list\_init** ()
- void **\_\_gnu\_profile::\_\_trace\_vector\_to\_list\_report** (FILE \*, \_\_warning\_vector\_t &)
- void **\_\_gnu\_profile::\_\_write\_cost\_factors** ()
- **\_\_gnu\_profile::\_\_GLIBCXX\_PROFILE\_DEFINE\_DATA** (\_\_trace\_hash\_func \*, \_\_S\_hash\_func, 0)
- **\_\_gnu\_profile::\_\_GLIBCXX\_PROFILE\_DEFINE\_DATA** (\_\_trace\_hashtable\_size \*, \_\_S\_hashtable\_size, 0)
- **\_\_gnu\_profile::\_\_GLIBCXX\_PROFILE\_DEFINE\_DATA** (\_\_trace\_map2umap \*, \_\_S\_map2umap, 0)
- **\_\_gnu\_profile::\_\_GLIBCXX\_PROFILE\_DEFINE\_DATA** (\_\_trace\_vector\_size \*, \_\_S\_vector\_size, 0)
- **\_\_gnu\_profile::\_\_GLIBCXX\_PROFILE\_DEFINE\_DATA** (\_\_trace\_vector\_to\_list \*, \_\_S\_vector\_to\_list, 0)
- **\_\_gnu\_profile::\_\_GLIBCXX\_PROFILE\_DEFINE\_DATA** (\_\_trace\_list\_to\_slist \*, \_\_S\_list\_to\_slist, 0)
- **\_\_gnu\_profile::\_\_GLIBCXX\_PROFILE\_DEFINE\_DATA** (\_\_trace\_list\_to\_vector \*, \_\_S\_list\_to\_vector, 0)
- **\_\_gnu\_profile::\_\_GLIBCXX\_PROFILE\_DEFINE\_DATA** (\_\_cost\_factor, \_\_vector\_shift\_cost\_factor, {"\_\_vector\_\_  
shift cost factor", 1.0})

- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__vector_iterate_cost_factor`,{"`__vector_iterate_cost_factor`", 1.0})
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__vector_resize_cost_factor`,{"`__vector_resize_cost_factor`", 1.0})
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__list_shift_cost_factor`,{"`__list_shift_cost_factor`", 0.0})
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__list_iterate_cost_factor`,{"`__list_iterate_cost_factor`", 10.0})
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__list_resize_cost_factor`,{"`__list_resize_cost_factor`", 0.0})
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__map_insert_cost_factor`,{"`__map_insert_cost_factor`", 1.5})
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__map_erase_cost_factor`,{"`__map_erase_cost_factor`", 1.5})
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__map_find_cost_factor`,{"`__map_find_cost_factor`", 1})
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__map_iterate_cost_factor`,{"`__map_iterate_cost_factor`", 2.3})
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__umap_insert_cost_factor`,{"`__umap_insert_cost_factor`", 12.0})
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__umap_erase_cost_factor`,{"`__umap_erase_cost_factor`", 12.0})
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__umap_find_cost_factor`,{"`__umap_find_cost_factor`", 10.0})
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__umap_iterate_cost_factor`,{"`__umap_iterate_cost_factor`", 1.7})
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor_vector` \*, `__cost_factors`, 0)
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA` (`const char` \*, `_S_trace_file_name`, `GLIBCXX_PROFILE_TRACE_PATH_ROOT`)
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA` (`std::size_t`, `_S_max_warn_count`, `GLIBCXX_PROFILE_MAX_WARN_COUNT`)
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA` (`std::size_t`, `_S_max_stack_depth`, `GLIBCXX_PROFILE_MAX_STACK_DEPTH`)
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA` (`std::size_t`, `_S_max_mem`, `GLIBCXX_PROFILE_MEMORY_PER_DIAGNOSTIC`)
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_UNINIT_DATA` (`__env_t`, `__env`)
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_UNINIT_DATA` (`__gnu_cxx::mutex`, `__global_lock`)

### 5.306.1 Detailed Description

Data structures to represent profiling traces.

## 5.307 profiler\_vector\_size.h File Reference

### Classes

- class `__gnu_profile::__trace_vector_size`

### Namespaces

- `__gnu_profile`

## Functions

- void `__gnu_profile::__trace_vector_size_construct` (const void \*, std::size\_t)
- void `__gnu_profile::__trace_vector_size_destruct` (const void \*, std::size\_t, std::size\_t)
- void `__gnu_profile::__trace_vector_size_init` ()
- void `__gnu_profile::__trace_vector_size_report` (FILE \*, \_\_warning\_vector\_t &)
- void `__gnu_profile::__trace_vector_size_resize` (const void \*, std::size\_t, std::size\_t)

## 5.307.1 Detailed Description

Collection of vector size traces.

## 5.308 profiler\_vector\_to\_list.h File Reference

## Classes

- class [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_to\\_list](#)
- class [\\_\\_gnu\\_profile::\\_\\_vector2list\\_info](#)
- class [\\_\\_gnu\\_profile::\\_\\_vector2list\\_stack\\_info](#)

## Namespaces

- [\\_\\_gnu\\_profile](#)

## Functions

- void `__gnu_profile::__trace_vector_to_list_construct` (const void \*)
- void `__gnu_profile::__trace_vector_to_list_destruct` (const void \*)
- void `__gnu_profile::__trace_vector_to_list_find` (const void \*, std::size\_t)
- void `__gnu_profile::__trace_vector_to_list_init` ()
- void `__gnu_profile::__trace_vector_to_list_insert` (const void \*, std::size\_t, std::size\_t)
- void `__gnu_profile::__trace_vector_to_list_invalid_operator` (const void \*)
- void `__gnu_profile::__trace_vector_to_list_iterate` (const void \*, std::size\_t)
- void `__gnu_profile::__trace_vector_to_list_report` (FILE \*, \_\_warning\_vector\_t &)
- void `__gnu_profile::__trace_vector_to_list_resize` (const void \*, std::size\_t, std::size\_t)

## 5.308.1 Detailed Description

diagnostics for vector to list.

## 5.309 ptr\_traits.h File Reference

## Classes

- struct [std::pointer\\_traits<\\_Ptr>](#)
- struct [std::pointer\\_traits<\\_Tp\\*>](#)

## Namespaces

- [std](#)

### 5.309.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 5.310 `quadratic_probe_fn_imp.hpp` File Reference

### 5.310.1 Detailed Description

Contains a probe policy implementation

## 5.311 `queue.h` File Reference

### Classes

- class [\\_\\_gnu\\_parallel::\\_RestrictedBoundedConcurrentQueue<\\_Tp>](#)

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Macros

- [#define \\_GLIBCXX\\_VOLATILE](#)

### 5.311.1 Detailed Description

Lock-free double-ended queue. This file is a GNU parallel extension to the Standard C++ Library.

### 5.311.2 Macro Definition Documentation

#### 5.311.2.1 [#define \\_GLIBCXX\\_VOLATILE](#)

Decide whether to declare certain variable volatile in this file.

Definition at line 40 of file `queue.h`.

## 5.312 `quicksort.h` File Reference

### Namespaces

- [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_sort_qs (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_sort_qs_conquer (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare> std::iterator_traits<_RAIter>::difference_type __gnu_parallel::__parallel_sort_qs_divide (_RAIter __begin, _RAIter __end, _Compare __comp, typename std::iterator_traits<_RAIter>::difference_type __pivot_rank, typename std::iterator_traits<_RAIter>::difference_type __num_samples, _ThreadIndex __num_threads)`

## 5.312.1 Detailed Description

Implementation of a unbalanced parallel quicksort (in-place). This file is a GNU parallel extension to the Standard C++ Library.

## 5.313 r\_erase\_fn\_imps.hpp File Reference

## 5.313.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

## 5.314 r\_erase\_fn\_imps.hpp File Reference

## 5.314.1 Detailed Description

Contains an implementation class for `pat_trie`.

## 5.315 random.h File Reference

## Classes

- class `std::bernoulli_distribution`
- struct `std::bernoulli_distribution::param_type`
- class `std::binomial_distribution<_IntType>`
- struct `std::binomial_distribution<_IntType>::param_type`
- class `std::cauchy_distribution<_RealType>`
- struct `std::cauchy_distribution<_RealType>::param_type`
- class `std::chi_squared_distribution<_RealType>`
- struct `std::chi_squared_distribution<_RealType>::param_type`
- class `std::discard_block_engine<_RandomNumberEngine, __p, __r>`
- class `std::discrete_distribution<_IntType>`
- struct `std::discrete_distribution<_IntType>::param_type`
- class `std::exponential_distribution<_RealType>`
- struct `std::exponential_distribution<_RealType>::param_type`
- class `std::extreme_value_distribution<_RealType>`
- struct `std::extreme_value_distribution<_RealType>::param_type`
- class `std::fisher_f_distribution<_RealType>`
- struct `std::fisher_f_distribution<_RealType>::param_type`

- class `std::gamma_distribution<_RealType>`
- struct `std::gamma_distribution<_RealType>::param_type`
- class `std::geometric_distribution<_IntType>`
- struct `std::geometric_distribution<_IntType>::param_type`
- class `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>`
- class `std::linear_congruential_engine<_UIntType, __a, __c, __m>`
- class `std::lognormal_distribution<_RealType>`
- struct `std::lognormal_distribution<_RealType>::param_type`
- class `std::mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f>`
- class `std::negative_binomial_distribution<_IntType>`
- struct `std::negative_binomial_distribution<_IntType>::param_type`
- class `std::normal_distribution<_RealType>`
- struct `std::normal_distribution<_RealType>::param_type`
- class `std::piecewise_constant_distribution<_RealType>`
- struct `std::piecewise_constant_distribution<_RealType>::param_type`
- class `std::piecewise_linear_distribution<_RealType>`
- struct `std::piecewise_linear_distribution<_RealType>::param_type`
- class `std::poisson_distribution<_IntType>`
- struct `std::poisson_distribution<_IntType>::param_type`
- class `std::random_device`
- class `std::seed_seq`
- class `std::shuffle_order_engine<_RandomNumberEngine, __k>`
- class `std::student_t_distribution<_RealType>`
- struct `std::student_t_distribution<_RealType>::param_type`
- class `std::uniform_int_distribution<_IntType>`
- struct `std::uniform_int_distribution<_IntType>::param_type`
- class `std::uniform_real_distribution<_RealType>`
- struct `std::uniform_real_distribution<_RealType>::param_type`
- class `std::weibull_distribution<_RealType>`
- struct `std::weibull_distribution<_RealType>::param_type`

## Namespaces

- `std`
- `std::__detail`

## Typedefs

- typedef `minstd_rand0` **`std::default_random_engine`**
- typedef `shuffle_order_engine< minstd_rand0, 256 >` **`std::knuth_b`**
- typedef `linear_congruential_engine< uint_fast32_t, 48271UL, 0UL, 2147483647UL >` `std::minstd_rand`
- typedef `linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL >` `std::minstd_rand0`
- typedef `mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL >` `std::mt19937`
- typedef `mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL >` `std::mt19937_64`
- typedef `discard_block_engine< ranlux24_base, 223, 23 >` **`std::ranlux24`**
- typedef `subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >` **`std::ranlux24_base`**
- typedef `discard_block_engine< ranlux48_base, 389, 11 >` **`std::ranlux48`**
- typedef `subtract_with_carry_engine< uint_fast64_t, 48, 5, 12 >` **`std::ranlux48_base`**

## Functions

- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator> _RealType std::generate_canonical(_UniformRandomNumberGenerator &__g)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> bool std::operator!= (const std::linear_congruential_engine<_UIntType, __a, __c, __m> &__lhs, const std::linear_congruential_engine<_UIntType, __a, __c, __m> &__rhs)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> bool std::operator!= (const std::mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f> &__lhs, const std::mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f> &__rhs)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r> bool std::operator!= (const std::subtract_with_carry_engine<_UIntType, __w, __s, __r> &__lhs, const std::subtract_with_carry_engine<_UIntType, __w, __s, __r> &__rhs)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r> bool std::operator!= (const std::discard_block_engine<_RandomNumberEngine, __p, __r> &__lhs, const std::discard_block_engine<_RandomNumberEngine, __p, __r> &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> bool std::operator!= (const std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType> &__lhs, const std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType> &__rhs)`
- `template<typename _RandomNumberEngine, size_t __k> bool std::operator!= (const std::shuffle_order_engine<_RandomNumberEngine, __k> &__lhs, const std::shuffle_order_engine<_RandomNumberEngine, __k> &__rhs)`
- `template<typename _IntType> bool std::operator!= (const std::uniform_int_distribution<_IntType> &__d1, const std::uniform_int_distribution<_IntType> &__d2)`
- `template<typename _IntType> bool std::operator!= (const std::uniform_real_distribution<_IntType> &__d1, const std::uniform_real_distribution<_IntType> &__d2)`
- `template<typename _RealType> bool std::operator!= (const std::normal_distribution<_RealType> &__d1, const std::normal_distribution<_RealType> &__d2)`
- `template<typename _RealType> bool std::operator!= (const std::lognormal_distribution<_RealType> &__d1, const std::lognormal_distribution<_RealType> &__d2)`
- `template<typename _RealType> bool std::operator!= (const std::gamma_distribution<_RealType> &__d1, const std::gamma_distribution<_RealType> &__d2)`
- `template<typename _RealType> bool std::operator!= (const std::chi_squared_distribution<_RealType> &__d1, const std::chi_squared_distribution<_RealType> &__d2)`
- `template<typename _RealType> bool std::operator!= (const std::cauchy_distribution<_RealType> &__d1, const std::cauchy_distribution<_RealType> &__d2)`
- `template<typename _RealType> bool std::operator!= (const std::fisher_f_distribution<_RealType> &__d1, const std::fisher_f_distribution<_RealType> &__d2)`
- `template<typename _RealType> bool std::operator!= (const std::student_t_distribution<_RealType> &__d1, const std::student_t_distribution<_RealType> &__d2)`
- `bool std::operator!= (const std::bernoulli_distribution &__d1, const std::bernoulli_distribution &__d2)`
- `template<typename _IntType> bool std::operator!= (const std::binomial_distribution<_IntType> &__d1, const std::binomial_distribution<_IntType> &__d2)`
- `template<typename _IntType> bool std::operator!= (const std::geometric_distribution<_IntType> &__d1, const std::geometric_distribution<_IntType> &__d2)`
- `template<typename _IntType> bool std::operator!= (const std::negative_binomial_distribution<_IntType> &__d1, const std::negative_binomial_distribution<_IntType> &__d2)`
- `template<typename _IntType> bool std::operator!= (const std::poisson_distribution<_IntType> &__d1, const std::poisson_distribution<_IntType> &__d2)`
- `template<typename _RealType> bool std::operator!= (const std::exponential_distribution<_RealType> &__d1, const std::exponential_distribution<_RealType> &__d2)`



- `template<typename _RealType > bool std::operator!= (const std::weibull_distribution< _RealType > &__d1, const std::weibull_distribution< _RealType > &__d2)`
- `template<typename _RealType > bool std::operator!= (const std::extreme_value_distribution< _RealType > &__d1, const std::extreme_value_distribution< _RealType > &__d2)`
- `template<typename _IntType > bool std::operator!= (const std::discrete_distribution< _IntType > &__d1, const std::discrete_distribution< _IntType > &__d2)`
- `template<typename _RealType > bool std::operator!= (const std::piecewise_constant_distribution< _RealType > &__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<typename _RealType > bool std::operator!= (const std::piecewise_linear_distribution< _RealType > &__d1, const std::piecewise_linear_distribution< _RealType > &__d2)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits > std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits > std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits > std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_real_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits > std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::cauchy_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits > std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::bernoulli_distribution &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits > std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::geometric_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits > std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits > std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits > std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits > std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits > std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits > std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::cauchy_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits > std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::bernoulli_distribution &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits > std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::geometric_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits > std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits > std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits > std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::extreme_value_distribution< _RealType > &__x)`

## 5.315.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<random>`.

## 5.316 random\_number.h File Reference

## Classes

- class [\\_\\_gnu\\_parallel::\\_RandomNumber](#)

## Namespaces

- [\\_\\_gnu\\_parallel](#)

## 5.316.1 Detailed Description

Random number generator based on the Mersenne twister. This file is a GNU parallel extension to the Standard C++ Library.

## 5.317 random\_shuffle.h File Reference

## Classes

- struct [\\_\\_gnu\\_parallel::\\_DRandomShufflingGlobalData<\\_RAIter>](#)
- struct [\\_\\_gnu\\_parallel::\\_DRSSorterPU<\\_RAIter, \\_RandomNumberGenerator>](#)

## Namespaces

- [\\_\\_gnu\\_parallel](#)

## Typedefs

- typedef unsigned short [\\_\\_gnu\\_parallel::\\_BinIndex](#)

## Functions

- template<typename \_RAIter, typename \_RandomNumberGenerator> void [\\_\\_gnu\\_parallel::\\_parallel\\_random\\_shuffle](#) (\_RAIter \_\_begin, \_RAIter \_\_end, \_RandomNumberGenerator \_\_rng=\_RandomNumber())
- template<typename \_RAIter, typename \_RandomNumberGenerator> void [\\_\\_gnu\\_parallel::\\_parallel\\_random\\_shuffle\\_drs](#) (\_RAIter \_\_begin, \_RAIter \_\_end, typename std::iterator\_traits<\_RAIter>::difference\_type \_\_n, \_ThreadIndex \_\_num\_threads, \_RandomNumberGenerator &\_\_rng)
- template<typename \_RAIter, typename \_RandomNumberGenerator> void [\\_\\_gnu\\_parallel::\\_parallel\\_random\\_shuffle\\_drs\\_←\\_pu](#) (\_DRSSorterPU<\_RAIter, \_RandomNumberGenerator> \*\_\_pus)
- template<typename \_RandomNumberGenerator> int [\\_\\_gnu\\_parallel::\\_random\\_number\\_pow2](#) (int \_\_logp, \_RandomNumberGenerator &\_\_rng)
- template<typename \_Tp> \_Tp [\\_\\_gnu\\_parallel::\\_round\\_up\\_to\\_pow2](#) (\_Tp \_\_x)
- template<typename \_RAIter, typename \_RandomNumberGenerator> void [\\_\\_gnu\\_parallel::\\_sequential\\_random\\_shuffle](#) (←\_RAIter \_\_begin, \_RAIter \_\_end, \_RandomNumberGenerator &\_\_rng)

### 5.317.1 Detailed Description

Parallel implementation of `std::random_shuffle()`. This file is a GNU parallel extension to the Standard C++ Library.

### 5.318 `range_access.h` File Reference

#### Namespaces

- [std](#)

#### Functions

- `template<class _Container > auto std::begin (_Container &__cont) -> decltype(__cont.begin())`
- `template<class _Container > auto std::begin (const _Container &__cont) -> decltype(__cont.begin())`
- `template<class _Tp, size_t _Nm> _Tp * std::begin (_Tp(&__arr)[_Nm])`
- `template<class _Container > auto std::end (_Container &__cont) -> decltype(__cont.end())`
- `template<class _Container > auto std::end (const _Container &__cont) -> decltype(__cont.end())`
- `template<class _Tp, size_t _Nm> _Tp * std::end (_Tp(&__arr)[_Nm])`

### 5.318.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

### 5.319 `ranged_hash_fn.hpp` File Reference

#### Classes

- `class \_\_gnu\_pbds::detail::ranged\_hash\_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >`
- `class \_\_gnu\_pbds::detail::ranged\_hash\_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false >`
- `class \_\_gnu\_pbds::detail::ranged\_hash\_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true >`
- `class \_\_gnu\_pbds::detail::ranged\_hash\_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false >`
- `class \_\_gnu\_pbds::detail::ranged\_hash\_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true >`

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### Macros

- `#define PB\_DS\_CLASS\_C\_DEC`
- `#define PB\_DS\_CLASS\_C\_DEC`
- `#define PB\_DS\_CLASS\_C\_DEC`
- `#define PB\_DS\_CLASS\_C\_DEC`
- `#define PB\_DS\_CLASS\_T\_DEC`
- `#define PB\_DS\_CLASS\_T\_DEC`
- `#define PB\_DS\_CLASS\_T\_DEC`
- `#define PB\_DS\_CLASS\_T\_DEC`

## 5.319.1 Detailed Description

Contains a unified ranged hash functor, allowing the hash tables to deal with a single class for ranged hashing.

5.320 `ranged_probe_fn.hpp` File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::ranged\\_probe\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Probe\\_Fn, Probe\\_Fn, Store\\_Hash >](#)
- class [\\_\\_gnu\\_pbds::detail::ranged\\_probe\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Probe\\_Fn, Probe\\_Fn, false >](#)
- class [\\_\\_gnu\\_pbds::detail::ranged\\_probe\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Probe\\_Fn, Probe\\_Fn, true >](#)
- class [\\_\\_gnu\\_pbds::detail::ranged\\_probe\\_fn< Key, null\\_type, \\_Alloc, Comb\\_Probe\\_Fn, null\\_type, false >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`

## 5.320.1 Detailed Description

Contains a unified ranged probe functor, allowing the probe tables to deal with a single class for ranged probing.

5.321 `rb_tree.hpp` File Reference

## Classes

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

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_RB_TREE_BASE`
- `#define PB_DS_RB_TREE_BASE_NAME`
- `#define PB_DS_RB_TREE_NAME`
- `#define PB_DS_STRUCT_ONLY_ASSERT_VALID(X)`

#### 5.321.1 Detailed Description

Contains an implementation for Red Black trees.

### 5.322 rc.hpp File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::detail::rc< \\_Node, \\_Alloc >](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.322.1 Detailed Description

Contains a redundant (binary counter).

### 5.323 rc\_binomial\_heap.hpp File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::detail::rc\\_binomial\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_RC_C_DEC`

#### 5.323.1 Detailed Description

Contains an implementation for redundant-counter binomial heap.

### 5.324 rc\_string\_base.h File Reference

#### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_rc\\_string\\_base< \\_CharT, \\_Traits, \\_Alloc >](#)

#### Namespaces

- [\\_\\_gnu\\_cxx](#)

## 5.324.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

## 5.325 regex.h File Reference

## Classes

- class [std::basic\\_regex](#)< [\\_Ch\\_type](#), [\\_Rx\\_traits](#) >
- class [std::match\\_results](#)< [\\_Bi\\_iter](#), [\\_Alloc](#) >
- class [std::regex\\_iterator](#)< [\\_Bi\\_iter](#), [\\_Ch\\_type](#), [\\_Rx\\_traits](#) >
- class [std::regex\\_token\\_iterator](#)< [\\_Bi\\_iter](#), [\\_Ch\\_type](#), [\\_Rx\\_traits](#) >
- struct [std::regex\\_traits](#)< [\\_Ch\\_type](#) >
- class [std::sub\\_match](#)< [\\_Biter](#) >

## Namespaces

- [std](#)

## Typedefs

- template<typename [\\_Bi\\_iter](#) , typename [\\_Ch\\_traits](#) , typename [\\_Ch\\_alloc](#) > using [std::\\_\\_sub\\_match\\_string](#) = [basic\\_string](#)< [typename iterator\\_traits](#)< [\\_Bi\\_iter](#) >::value\_type, [\\_Ch\\_traits](#), [\\_Ch\\_alloc](#) >
- typedef [match\\_results](#)< const char \* > [std::cmatch](#)
- typedef [regex\\_iterator](#)< const char \* > [std::cregex\\_iterator](#)
- typedef [regex\\_token\\_iterator](#)< const char \* > [std::cregex\\_token\\_iterator](#)
- typedef [sub\\_match](#)< const char \* > [std::csub\\_match](#)
- typedef [basic\\_regex](#)< char > [std::regex](#)
- typedef [match\\_results](#)< [string::const\\_iterator](#) > [std::smatch](#)
- typedef [regex\\_iterator](#)< [string::const\\_iterator](#) > [std::sregex\\_iterator](#)
- typedef [regex\\_token\\_iterator](#)< [string::const\\_iterator](#) > [std::sregex\\_token\\_iterator](#)
- typedef [sub\\_match](#)< [string::const\\_iterator](#) > [std::ssub\\_match](#)
- typedef [match\\_results](#)< const [wchar\\_t](#) \* > [std::wcmatch](#)
- typedef [regex\\_iterator](#)< const [wchar\\_t](#) \* > [std::wcregex\\_iterator](#)
- typedef [regex\\_token\\_iterator](#)< const [wchar\\_t](#) \* > [std::wcregex\\_token\\_iterator](#)
- typedef [sub\\_match](#)< const [wchar\\_t](#) \* > [std::wcs\\_sub\\_match](#)
- typedef [basic\\_regex](#)< [wchar\\_t](#) > [std::wregex](#)
- typedef [match\\_results](#)< [wstring::const\\_iterator](#) > [std::wsmatch](#)
- typedef [regex\\_iterator](#)< [wstring::const\\_iterator](#) > [std::wsregex\\_iterator](#)
- typedef [regex\\_token\\_iterator](#)< [wstring::const\\_iterator](#) > [std::wsregex\\_token\\_iterator](#)
- typedef [sub\\_match](#)< [wstring::const\\_iterator](#) > [std::wssub\\_match](#)

## Functions

- template<typename [\\_Bi\\_iter](#) > const [sub\\_match](#)< [\\_Bi\\_iter](#) > & [std::\\_\\_unmatched\\_sub](#) ()
- template<typename [\\_Biter](#) > bool [std::operator!=](#) (const [sub\\_match](#)< [\\_Biter](#) > &\_\_lhs, const [sub\\_match](#)< [\\_Biter](#) > &\_\_rhs)
- template<typename [\\_Bi\\_iter](#) , typename [\\_Ch\\_traits](#) , typename [\\_Ch\\_alloc](#) > bool [std::operator!=](#) (const [\\_\\_sub\\_match\\_string](#)< [\\_Bi\\_iter](#), [\\_Ch\\_traits](#), [\\_Ch\\_alloc](#) > &\_\_lhs, const [sub\\_match](#)< [\\_Bi\\_iter](#) > &\_\_rhs)

- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> bool std::operator!= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter> bool std::operator!= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter> bool std::operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter> bool std::operator!= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter> bool std::operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter, class _Alloc> bool std::operator!= (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)`
- `template<typename _Bilter> bool std::operator< (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> bool std::operator< (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc> bool std::operator< (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter> bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter> bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter> bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter> bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter> basic_ostream< _Ch_type, _Ch_traits > & std::operator<< (basic_ostream< _Ch_type, _Ch_traits > &__os, const sub_match< _Bi_iter > &__m)`
- `template<typename _Bilter> bool std::operator<= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> bool std::operator<= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc> bool std::operator<= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter> bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter> bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter> bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter> bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bilter> bool std::operator== (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> bool std::operator== (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> bool std::operator== (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter> bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter> bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`

- `template<typename _Bi_iter > bool std::operator==(typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter > bool std::operator==(const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter, typename _Alloc > bool std::operator==(const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)`
- `template<typename _Bilter > bool std::operator> (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc > bool std::operator> (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc > bool std::operator> (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter > bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter > bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter > bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter > bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bilter > bool std::operator>= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc > bool std::operator>= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc > bool std::operator>= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter > bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter > bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter > bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter > bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Ch_type, typename _Rx_traits > void std::swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _Ch_type, _Rx_traits > &__rhs)`
- `template<typename _Bi_iter, typename _Alloc > void std::swap (match_results< _Bi_iter, _Alloc > &__lhs, match_results< _Bi_iter, _Alloc > &__rhs)`

### Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits > bool std::regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits > bool std::regex_match (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Alloc, typename _Rx_traits > bool std::regex_match (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits > bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`



- `template<typename _Ch_type, class _Rx_traits> bool std::regex\_match (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits> &__re, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits> bool std::regex\_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator> &__s, const basic_regex< _Ch_type, _Rx_traits> &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits> bool std::regex\_search (_Bi_iter __first, _Bi_iter __last, match_results< _Bi_iter, _Alloc> &__m, const basic_regex< _Ch_type, _Rx_traits> &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits> bool std::regex\_search (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits> &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Alloc, class _Rx_traits> bool std::regex\_search (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc> &__m, const basic_regex< _Ch_type, _Rx_traits> &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Rx_traits> bool std::regex\_search (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits> &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits> bool std::regex\_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator> &__s, const basic_regex< _Ch_type, _Rx_traits> &__e, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits> bool std::regex\_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc> &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc>::const_iterator, _Alloc> &__m, const basic_regex< _Ch_type, _Rx_traits> &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type> _Out_iter std::regex\_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits> &__e, const basic_string< _Ch_type> &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type> basic_string< _Ch_type> std::regex\_replace (const basic_string< _Ch_type> &__s, const basic_regex< _Ch_type, _Rx_traits> &__e, const basic_string< _Ch_type> &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`

### 5.325.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 5.326 `regex_compiler.h` File Reference

### Classes

- class [std::\\_\\_detail::\\_\\_Compiler<\\_InIter, \\_TraitsT>](#)
- class [std::\\_\\_detail::\\_\\_Scanner<\\_InputIterator>](#)
- struct [std::\\_\\_detail::\\_\\_Scanner\\_base](#)

### Namespaces

- [std](#)
- [std::\\_\\_detail](#)

## Functions

- `template<typename _InIter, typename _TraitsT> _AutomatonPtr std::__detail::__compile (const _InIter &__b, const _InIter &__e, _TraitsT &__t, regex_constants::syntax_option_type __f)`

## 5.326.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

5.327 `regex_constants.h` File Reference

## Namespaces

- [std](#)
- [std::regex\\_constants](#)

## 5.1 Regular Expression Syntax Options

- enum [std::regex\\_constants::\\_\\_syntax\\_option](#) { [\\_S\\_icode](#), [\\_S\\_nosubs](#), [\\_S\\_optimize](#), [\\_S\\_collate](#), [\\_S\\_ECMA←Script](#), [\\_S\\_basic](#), [\\_S\\_extended](#), [\\_S\\_awk](#), [\\_S\\_grep](#), [\\_S\\_egrep](#), [\\_S\\_syntax\\_last](#) }
- typedef unsigned int [std::regex\\_constants::syntax\\_option\\_type](#)
- constexpr syntax\_option\_type [std::regex\\_constants::icase](#)
- constexpr syntax\_option\_type [std::regex\\_constants::nosubs](#)
- constexpr syntax\_option\_type [std::regex\\_constants::optimize](#)
- constexpr syntax\_option\_type [std::regex\\_constants::collate](#)
- constexpr syntax\_option\_type [std::regex\\_constants::ECMAScript](#)
- constexpr syntax\_option\_type [std::regex\\_constants::basic](#)
- constexpr syntax\_option\_type [std::regex\\_constants::extended](#)
- constexpr syntax\_option\_type [std::regex\\_constants::awk](#)
- constexpr syntax\_option\_type [std::regex\\_constants::grep](#)
- constexpr syntax\_option\_type [std::regex\\_constants::egrep](#)

## 5.2 Matching Rules

Matching a regular expression against a sequence of characters [first, last) proceeds according to the rules of the grammar specified for the regular expression object, modified according to the effects listed below for any bitmask elements set.

- enum [std::regex\\_constants::\\_\\_match\\_flag](#) { [\\_S\\_not\\_bol](#), [\\_S\\_not\\_eol](#), [\\_S\\_not\\_bow](#), [\\_S\\_not\\_eow](#), [\\_S\\_any, \\_←S\\_not\\_null](#), [\\_S\\_continuous](#), [\\_S\\_prev\\_avail](#), [\\_S\\_sed](#), [\\_S\\_no\\_copy](#), [\\_S\\_first\\_only](#), [\\_S\\_match\\_flag\\_last](#) }
- typedef std::bitset< [\\_S\\_match\\_flag\\_last](#) > [std::regex\\_constants::match\\_flag\\_type](#)
- constexpr match\_flag\_type [std::regex\\_constants::match\\_default](#)
- constexpr match\_flag\_type [std::regex\\_constants::match\\_not\\_bol](#)
- constexpr match\_flag\_type [std::regex\\_constants::match\\_not\\_eol](#)
- constexpr match\_flag\_type [std::regex\\_constants::match\\_not\\_bow](#)
- constexpr match\_flag\_type [std::regex\\_constants::match\\_not\\_eow](#)
- constexpr match\_flag\_type [std::regex\\_constants::match\\_any](#)
- constexpr match\_flag\_type [std::regex\\_constants::match\\_not\\_null](#)
- constexpr match\_flag\_type [std::regex\\_constants::match\\_continuous](#)

- constexpr match\_flag\_type [std::regex\\_constants::match\\_prev\\_avail](#)
- constexpr match\_flag\_type [std::regex\\_constants::format\\_default](#)
- constexpr match\_flag\_type [std::regex\\_constants::format\\_sed](#)
- constexpr match\_flag\_type [std::regex\\_constants::format\\_no\\_copy](#)
- constexpr match\_flag\_type [std::regex\\_constants::format\\_first\\_only](#)

#### 5.327.1 Detailed Description

Constant definitions for the std regex library.

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

### 5.328 `regex_cursor.h` File Reference

#### Classes

- struct [std::\\_\\_detail::\\_PatternCursor](#)
- class [std::\\_\\_detail::\\_SpecializedCursor<\\_FwdIterT>](#)

#### Namespaces

- [std](#)
- [std::\\_\\_detail](#)

#### Functions

- template<typename \_FwdIterT> [\\_SpecializedCursor<\\_FwdIterT>](#) **std::\_\_detail::\_\_cursor** (const \_FwdIterT &\_\_b, const \_FwdIterT \_\_e)

#### 5.328.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

### 5.329 `regex_error.h` File Reference

#### Classes

- class [std::regex\\_error](#)

#### Namespaces

- [std](#)
- [std::regex\\_constants](#)

#### Functions

- void **std::\_\_throw\_regex\_error** (regex\_constants::error\_type \_\_ecode)

### 5.3 Error Types

- enum [std::regex\\_constants::error\\_type](#) { [\\_S\\_error\\_collate](#), [\\_S\\_error\\_ctype](#), [\\_S\\_error\\_escape](#), [\\_S\\_error\\_backref](#), [\\_S\\_error\\_brack](#), [\\_S\\_error\\_paren](#), [\\_S\\_error\\_brace](#), [\\_S\\_error\\_badbrace](#), [\\_S\\_error\\_range](#), [\\_S\\_error\\_space](#), [\\_S\\_error\\_badrepeat](#), [\\_S\\_error\\_complexity](#), [\\_S\\_error\\_stack](#), [\\_S\\_error\\_last](#) }
- constexpr error\_type [std::regex\\_constants::error\\_collate](#) ([\\_S\\_error\\_collate](#))
- constexpr error\_type [std::regex\\_constants::error\\_ctype](#) ([\\_S\\_error\\_ctype](#))
- constexpr error\_type [std::regex\\_constants::error\\_escape](#) ([\\_S\\_error\\_escape](#))
- constexpr error\_type [std::regex\\_constants::error\\_backref](#) ([\\_S\\_error\\_backref](#))
- constexpr error\_type [std::regex\\_constants::error\\_brack](#) ([\\_S\\_error\\_brack](#))
- constexpr error\_type [std::regex\\_constants::error\\_paren](#) ([\\_S\\_error\\_paren](#))
- constexpr error\_type [std::regex\\_constants::error\\_brace](#) ([\\_S\\_error\\_brace](#))
- constexpr error\_type [std::regex\\_constants::error\\_badbrace](#) ([\\_S\\_error\\_badbrace](#))
- constexpr error\_type [std::regex\\_constants::error\\_range](#) ([\\_S\\_error\\_range](#))
- constexpr error\_type [std::regex\\_constants::error\\_space](#) ([\\_S\\_error\\_space](#))
- constexpr error\_type [std::regex\\_constants::error\\_badrepeat](#) ([\\_S\\_error\\_badrepeat](#))
- constexpr error\_type [std::regex\\_constants::error\\_complexity](#) ([\\_S\\_error\\_complexity](#))
- constexpr error\_type [std::regex\\_constants::error\\_stack](#) ([\\_S\\_error\\_stack](#))

#### 5.329.1 Detailed Description

Error and exception objects for the std regex library.

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

### 5.330 regex\_grep\_matcher.h File Reference

#### Classes

- class [std::\\_\\_detail::\\_Grep\\_matcher](#)
- class [std::\\_\\_detail::\\_SpecializedResults<\\_FwdIterT, \\_Alloc>](#)
- class [std::match\\_results<\\_Bi\\_iter, \\_Alloc>](#)
- class [std::sub\\_match<\\_Biter>](#)

#### Namespaces

- [std](#)
- [std::\\_\\_detail](#)

#### Typedefs

- typedef [std::stack<\\_StateIdT, std::vector<\\_StateIdT>>](#) [std::\\_\\_detail::\\_StateStack](#)

#### 5.330.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 5.331 `regex_nfa.h` File Reference

### Classes

- class `std::__detail::_Automaton`
- struct `std::__detail::_CharMatcher<_InIterT, _TraitsT>`
- struct `std::__detail::_EndTagger<_FwdIterT, _TraitsT>`
- class `std::__detail::_Nfa`
- struct `std::__detail::_RangeMatcher<_InIterT, _TraitsT>`
- struct `std::__detail::_Results`
- struct `std::__detail::_StartTagger<_FwdIterT, _TraitsT>`
- struct `std::__detail::_State`
- class `std::__detail::_StateSeq`

### Namespaces

- `std`
- `std::__detail`

### Typedefs

- typedef `std::shared_ptr<_Automaton>` `std::__detail::_AutomatonPtr`
- typedef `std::function<bool(const _PatternCursor &)>` `std::__detail::_Matcher`
- typedef `int` `std::__detail::_StateIdT`
- typedef `std::set<_StateIdT>` `std::__detail::_StateSet`
- typedef `std::function<void(const _PatternCursor &, _Results &)>` `std::__detail::_Tagger`

### Enumerations

- enum `std::__detail::_Opcode` { `_S_opcode_unknown`, `_S_opcode_alternative`, `_S_opcode_subexpr_begin`, `_S_opcode_subexpr_end`, `_S_opcode_match`, `_S_opcode_accept` }

### Functions

- bool `std::__detail::_AnyMatcher` (const `_PatternCursor` &)

### Variables

- static const `_StateIdT` `std::__detail::_S_invalid_state_id`

### 5.331.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 5.332 `resize_fn_imps.hpp` File Reference

### 5.332.1 Detailed Description

Contains implementations of `cc_ht_map_`'s resize related functions.

### 5.333 `resize_fn_imps.hpp` File Reference

#### 5.333.1 Detailed Description

Contains implementations of `gp_ht_map_`'s resize related functions.

### 5.334 `resize_no_store_hash_fn_imps.hpp` File Reference

#### 5.334.1 Detailed Description

Contains implementations of `cc_ht_map_`'s resize related functions, when the hash value is not stored.

### 5.335 `resize_no_store_hash_fn_imps.hpp` File Reference

#### 5.335.1 Detailed Description

Contains implementations of `gp_ht_map_`'s resize related functions, when the hash value is not stored.

### 5.336 `resize_policy.hpp` File Reference

#### Classes

- class [`\_\_gnu\_pbds::detail::resize\_policy< \_Tp >`](#)

#### Namespaces

- [`\_\_gnu\_pbds`](#)

#### 5.336.1 Detailed Description

Contains an implementation class for a `binary_heap`.

### 5.337 `resize_store_hash_fn_imps.hpp` File Reference

#### 5.337.1 Detailed Description

Contains implementations of `cc_ht_map_`'s resize related functions, when the hash value is stored.

### 5.338 `resize_store_hash_fn_imps.hpp` File Reference

#### 5.338.1 Detailed Description

Contains implementations of `gp_ht_map_`'s resize related functions, when the hash value is stored.

## 5.339 ropeimpl.h File Reference

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### Functions

- `template<class _CharT, class _Traits> void __gnu_cxx::Rope_fill (basic_ostream< _CharT, _Traits> &__o, size_t __n)`
- `template<class _CharT> bool __gnu_cxx::Rope_is_simple (_CharT *)`
- `bool __gnu_cxx::Rope_is_simple (char *)`
- `bool __gnu_cxx::Rope_is_simple (wchar_t *)`
- `template<class _Rope_iterator> void __gnu_cxx::Rope_rotate (_Rope_iterator __first, _Rope_iterator __middle, _Rope_iterator __last)`
- `template<class _CharT, class _Traits, class _Alloc> basic_ostream< _CharT, _Traits> & __gnu_cxx::operator<< (basic_ostream< _CharT, _Traits> &__o, const rope< _CharT, _Alloc> &__r)`
- `void __gnu_cxx::rotate (_Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __first, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __middle, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __last)`

### 5.339.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/rope>`.

## 5.340 rotate\_fn\_imps.hpp File Reference

### 5.340.1 Detailed Description

Contains imps for rotating nodes.

## 5.341 rotate\_fn\_imps.hpp File Reference

### 5.341.1 Detailed Description

Contains imps for rotating nodes.

## 5.342 safe\_base.h File Reference

### Classes

- class [\\_\\_gnu\\_debug::\\_Safe\\_iterator\\_base](#)
- class [\\_\\_gnu\\_debug::\\_Safe\\_sequence\\_base](#)

### Namespaces

- [\\_\\_gnu\\_debug](#)

## 5.342.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

5.343 `safe_iterator.h` File Reference

## Classes

- struct [\\_\\_gnu\\_debug::\\_BeforeBeginHelper<\\_Sequence>](#)
- class [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Iterator, \\_Sequence>](#)

## Namespaces

- [\\_\\_gnu\\_debug](#)

## Enumerations

- enum [\\_\\_gnu\\_debug::Distance\\_precision](#) { `__dp_equality`, `__dp_sign`, `__dp_exact` }

## Functions

- bool [\\_\\_gnu\\_debug::\\_\\_check\\_singular\\_aux](#) (const `_Safe_iterator_base` \* \_\_x)
- template<typename `_Iterator1`, typename `_Iterator2`> [std::pair](#)< typename `std::iterator_traits`< `_Iterator1` >::difference\_type, `_Distance_precision` > [\\_\\_gnu\\_debug::\\_\\_get\\_distance](#) (const `_Iterator1` & \_\_lhs, const `_Iterator2` & \_\_rhs, [std::random\\_access\\_iterator\\_tag](#))
- template<typename `_Iterator1`, typename `_Iterator2`> [std::pair](#)< typename `std::iterator_traits`< `_Iterator1` >::difference\_type, `_Distance_precision` > [\\_\\_gnu\\_debug::\\_\\_get\\_distance](#) (const `_Iterator1` & \_\_lhs, const `_Iterator2` & \_\_rhs, [std::forward\\_iterator\\_tag](#))
- template<typename `_Iterator1`, typename `_Iterator2`> [std::pair](#)< typename `std::iterator_traits`< `_Iterator1` >::difference\_type, `_Distance_precision` > [\\_\\_gnu\\_debug::\\_\\_get\\_distance](#) (const `_Iterator1` & \_\_lhs, const `_Iterator2` & \_\_rhs)
- template<typename `_IteratorL`, typename `_IteratorR`, typename `_Sequence`> bool [\\_\\_gnu\\_debug::operator!=](#) (const `_Safe_iterator`< `_IteratorL`, `_Sequence` > & \_\_lhs, const `_Safe_iterator`< `_IteratorR`, `_Sequence` > & \_\_rhs)
- template<typename `_Iterator`, typename `_Sequence`> bool [\\_\\_gnu\\_debug::operator!=](#) (const `_Safe_iterator`< `_Iterator`, `_Sequence` > & \_\_lhs, const `_Safe_iterator`< `_Iterator`, `_Sequence` > & \_\_rhs)
- template<typename `_Iterator`, typename `_Sequence`> `_Safe_iterator`< `_Iterator`, `_Sequence` > [\\_\\_gnu\\_debug::operator+](#) (typename `_Safe_iterator`< `_Iterator`, `_Sequence` >::difference\_type \_\_n, const `_Safe_iterator`< `_Iterator`, `_Sequence` > & \_\_i)
- template<typename `_IteratorL`, typename `_IteratorR`, typename `_Sequence`> `_Safe_iterator`< `_IteratorL`, `_Sequence` >::difference\_type [\\_\\_gnu\\_debug::operator-](#) (const `_Safe_iterator`< `_IteratorL`, `_Sequence` > & \_\_lhs, const `_Safe_iterator`< `_IteratorR`, `_Sequence` > & \_\_rhs)
- template<typename `_Iterator`, typename `_Sequence`> `_Safe_iterator`< `_Iterator`, `_Sequence` >::difference\_type [\\_\\_gnu\\_debug::operator-](#) (const `_Safe_iterator`< `_Iterator`, `_Sequence` > & \_\_lhs, const `_Safe_iterator`< `_Iterator`, `_Sequence` > & \_\_rhs)
- template<typename `_IteratorL`, typename `_IteratorR`, typename `_Sequence`> bool [\\_\\_gnu\\_debug::operator<](#) (const `_Safe_iterator`< `_IteratorL`, `_Sequence` > & \_\_lhs, const `_Safe_iterator`< `_IteratorR`, `_Sequence` > & \_\_rhs)
- template<typename `_Iterator`, typename `_Sequence`> bool [\\_\\_gnu\\_debug::operator<](#) (const `_Safe_iterator`< `_Iterator`, `_Sequence` > & \_\_lhs, const `_Safe_iterator`< `_Iterator`, `_Sequence` > & \_\_rhs)
- template<typename `_IteratorL`, typename `_IteratorR`, typename `_Sequence`> bool [\\_\\_gnu\\_debug::operator<=](#) (const `_Safe_iterator`< `_IteratorL`, `_Sequence` > & \_\_lhs, const `_Safe_iterator`< `_IteratorR`, `_Sequence` > & \_\_rhs)
- template<typename `_Iterator`, typename `_Sequence`> bool [\\_\\_gnu\\_debug::operator<=](#) (const `_Safe_iterator`< `_Iterator`, `_Sequence` > & \_\_lhs, const `_Safe_iterator`< `_Iterator`, `_Sequence` > & \_\_rhs)



- `template<typename _IteratorL, typename _IteratorR, typename _Sequence> bool __gnu_debug::operator==(const _Safe_iterator< _IteratorL, _Sequence> &__lhs, const _Safe_iterator< _IteratorR, _Sequence> &__rhs)`
- `template<typename _Iterator, typename _Sequence> bool __gnu_debug::operator==(const _Safe_iterator< _Iterator, _Sequence> &__lhs, const _Safe_iterator< _Iterator, _Sequence> &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence> bool __gnu_debug::operator> (const _Safe_iterator< _IteratorL, _Sequence> &__lhs, const _Safe_iterator< _IteratorR, _Sequence> &__rhs)`
- `template<typename _Iterator, typename _Sequence> bool __gnu_debug::operator> (const _Safe_iterator< _Iterator, _Sequence> &__lhs, const _Safe_iterator< _Iterator, _Sequence> &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence> bool __gnu_debug::operator>= (const _Safe_iterator< _IteratorL, _Sequence> &__lhs, const _Safe_iterator< _IteratorR, _Sequence> &__rhs)`
- `template<typename _Iterator, typename _Sequence> bool __gnu_debug::operator>= (const _Safe_iterator< _Iterator, _Sequence> &__lhs, const _Safe_iterator< _Iterator, _Sequence> &__rhs)`

#### 5.343.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

### 5.344 `safe_local_iterator.h` File Reference

#### Classes

- class [`\_\_gnu\_debug::Safe\_local\_iterator< \_Iterator, \_Sequence>`](#)

#### Namespaces

- [`\_\_gnu\_debug`](#)

#### Functions

- `template<typename _IteratorL, typename _IteratorR, typename _Sequence> bool __gnu_debug::operator!=(const _Safe_local_iterator< _IteratorL, _Sequence> &__lhs, const _Safe_local_iterator< _IteratorR, _Sequence> &__rhs)`
- `template<typename _Iterator, typename _Sequence> bool __gnu_debug::operator!=(const _Safe_local_iterator< _Iterator, _Sequence> &__lhs, const _Safe_local_iterator< _Iterator, _Sequence> &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence> bool __gnu_debug::operator==(const _Safe_local_iterator< _IteratorL, _Sequence> &__lhs, const _Safe_local_iterator< _IteratorR, _Sequence> &__rhs)`
- `template<typename _Iterator, typename _Sequence> bool __gnu_debug::operator==(const _Safe_local_iterator< _Iterator, _Sequence> &__lhs, const _Safe_local_iterator< _Iterator, _Sequence> &__rhs)`

#### 5.344.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

### 5.345 `safe_sequence.h` File Reference

#### Classes

- class [`\_\_gnu\_debug::After\_nth\_from< \_Iterator>`](#)
- class [`\_\_gnu\_debug::Equal\_to< \_Type>`](#)
- class [`\_\_gnu\_debug::Not\_equal\_to< \_Type>`](#)

- class [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Iterator, \\_Sequence >](#)
- class [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence >](#)

#### Namespaces

- [\\_\\_gnu\\_debug](#)

#### 5.345.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

### 5.346 `safe_unordered_base.h` File Reference

#### Classes

- class [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator\\_base](#)
- class [\\_\\_gnu\\_debug::\\_Safe\\_unordered\\_container\\_base](#)

#### Namespaces

- [\\_\\_gnu\\_debug](#)

#### 5.346.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

### 5.347 `safe_unordered_container.h` File Reference

#### Classes

- class [\\_\\_gnu\\_debug::\\_Safe\\_unordered\\_container<\\_Container >](#)

#### Namespaces

- [\\_\\_gnu\\_debug](#)

#### 5.347.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

### 5.348 `sample_probe_fn.hpp` File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::sample\\_probe\\_fn](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

### 5.348.1 Detailed Description

Contains a sample probe policy.

## 5.349 sample\_range\_hashing.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::sample\\_range\\_hashing](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

### 5.349.1 Detailed Description

Contains a range hashing policy.

## 5.350 sample\_ranged\_hash\_fn.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::sample\\_ranged\\_hash\\_fn](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

### 5.350.1 Detailed Description

Contains a ranged hash policy.

## 5.351 sample\_ranged\_probe\_fn.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::sample\\_ranged\\_probe\\_fn](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## 5.351.1 Detailed Description

Contains a ranged probe policy.

5.352 `sample_resize_policy.hpp` File Reference

## Classes

- class [\\_\\_gnu\\_pbds::sample\\_resize\\_policy](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## 5.352.1 Detailed Description

Contains a sample resize policy for hash tables.

5.353 `sample_resize_trigger.hpp` File Reference

## Classes

- class [\\_\\_gnu\\_pbds::sample\\_resize\\_trigger](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## 5.353.1 Detailed Description

Contains a sample resize trigger policy class.

5.354 `sample_size_policy.hpp` File Reference

## Classes

- class [\\_\\_gnu\\_pbds::sample\\_size\\_policy](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## 5.354.1 Detailed Description

Contains a sample size resize-policy.

### 5.355 `sample_tree_node_update.hpp` File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::sample\\_tree\\_node\\_update](#)< `Const_Node_Iter`, `Node_Iter`, `Cmp_Fn`, `_Alloc` >

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.355.1 Detailed Description

Contains a samle node update functor.

### 5.356 `sample_trie_access_traits.hpp` File Reference

#### Classes

- struct [\\_\\_gnu\\_pbds::sample\\_trie\\_access\\_traits](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.356.1 Detailed Description

Contains a sample probe policy.

### 5.357 `sample_trie_node_update.hpp` File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::sample\\_trie\\_node\\_update](#)< `Node_Cltr`, `Node_Itr`, `_ATraits`, `_Alloc` >

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.357.1 Detailed Description

Contains a samle node update functor.

### 5.358 `sample_update_policy.hpp` File Reference

#### Classes

- struct [\\_\\_gnu\\_pbds::sample\\_update\\_policy](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## 5.358.1 Detailed Description

Contains a sample policy for list update containers.

## 5.359 search.h File Reference

## Namespaces

- [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _RAIter, typename _DifferenceTp > void \_\_gnu\_parallel::\_\_calc\_borders (_RAIter __elements, ↵ _DifferenceTp __length, _DifferenceTp *__off)`
- `template<typename __RAIter1, typename __RAIter2, typename _Pred > __RAIter1 \_\_gnu\_parallel::\_\_search\_template (↵ __RAIter1 __begin1, __RAIter1 __end1, __RAIter2 __begin2, __RAIter2 __end2, _Pred __pred)`

## 5.359.1 Detailed Description

Parallel implementation base for `std::search()` and `std::search_n()`. This file is a GNU parallel extension to the Standard C++ Library.

## 5.360 set.h File Reference

## Classes

- class `std::__debug::set<_Key, _Compare, _Allocator >`

## Namespaces

- [std](#)
- [std::\\_\\_debug](#)

## Functions

- `template<typename _Key, typename _Compare, typename _Allocator > bool std::__debug::operator!= (const set<_Key, _Compare, _Allocator > &__lhs, const set<_Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator > bool std::__debug::operator< (const set<_Key, _Compare, _Allocator > &__lhs, const set<_Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator > bool std::__debug::operator<= (const set<_Key, _Compare, _Allocator > &__lhs, const set<_Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator > bool std::__debug::operator== (const set<_Key, _Compare, _Allocator > &__lhs, const set<_Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator > bool std::__debug::operator> (const set<_Key, _Compare, _Allocator > &__lhs, const set<_Key, _Compare, _Allocator > &__rhs)`

- `template<typename _Key , typename _Compare , typename _Allocator > bool std::__debug::operator>= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator > void std::__debug::swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y)`

#### 5.360.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

### 5.361 `set.h` File Reference

#### Classes

- class `std::\_\_profile::set< \_Key, \_Compare, \_Allocator >`

#### Namespaces

- `std`
- `std::\_\_profile`

#### Functions

- `template<typename _Key , typename _Compare , typename _Allocator > bool std::__profile::operator!= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator > bool std::__profile::operator< (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator > bool std::__profile::operator<= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator > bool std::__profile::operator== (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator > bool std::__profile::operator> (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator > bool std::__profile::operator>= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator > void std::__profile::swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y)`

#### 5.361.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

### 5.362 `set_operations.h` File Reference

#### Namespaces

- `\_\_gnu\_parallel`

## Functions

- `template<typename _Iter, typename _OutputIterator> _OutputIterator __gnu_parallel::__copy_tail (std::pair<_Iter, ↵  
_Iter> __b, std::pair<_Iter, _Iter> __e, _OutputIterator __r)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare> _OutputIterator __gnu_parallel::__parallel↵  
__set_difference (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, ↵  
_Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare> _OutputIterator __gnu_parallel::__parallel↵  
__set_intersection (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, ↵  
_Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Operation> _OutputIterator __gnu_parallel::__parallel↵  
__set_operation (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, ↵  
_Operation __op)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare> _OutputIterator __gnu_parallel::__parallel↵  
__set_symmetric_difference (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __↵  
result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare> _OutputIterator __gnu_parallel::__parallel↵  
__set_union (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare ↵  
__comp)`

## 5.362.1 Detailed Description

Parallel implementations of set operations for random-access iterators. This file is a GNU parallel extension to the Standard C++ Library.

## 5.363 settings.h File Reference

## Classes

- struct [\\_\\_gnu\\_parallel::Settings](#)

## Namespaces

- [\\_\\_gnu\\_parallel](#)

## Macros

- `#define \_GLIBCXX\_PARALLEL\_CONDITION(__c)`

## 5.363.1 Detailed Description

Runtime settings and tuning parameters, heuristics to decide whether to use parallelized algorithms. This file is a GNU parallel extension to the Standard C++ Library.

## 5.363.2 parallelization\_decision

The decision whether to run an algorithm in parallel.

There are several ways the user can switch on and \_\_off the parallel execution of an algorithm, both at compile- and run-time.



Only sequential execution can be forced at compile-time. This reduces code size and protects code parts that have non-thread-safe side effects.

Ultimately, forcing parallel execution at compile-time makes sense. Often, the sequential algorithm implementation is used as a subroutine, so no reduction in code size can be achieved. Also, the machine the program is run on might have only one processor core, so to avoid overhead, the algorithm is executed sequentially.

To force sequential execution of an algorithm ultimately at compile-time, the user must add the tag `gnu_parallel::sequential_tag()` to the end of the parameter list, e. g.

```
std::sort (__v.begin(), __v.end(), __gnu_parallel::sequential_tag());
```

This is compatible with all overloaded algorithm variants. No additional code will be instantiated, at all. The same holds for most algorithm calls with iterators not providing random access.

If the algorithm call is not forced to be executed sequentially at compile-time, the decision is made at run-time. The global variable `__gnu_parallel::_Settings::algorithm_strategy` is checked. It is a tristate variable corresponding to:

a. `force_sequential`, meaning the sequential algorithm is executed. b. `force_parallel`, meaning the parallel algorithm is executed. c. `heuristic`

For heuristic, the parallel algorithm implementation is called only if the input size is sufficiently large. For most algorithms, the input size is the (combined) length of the input sequence(`__s`). The threshold can be set by the user, individually for each algorithm. The according variables are called `gnu_parallel::_Settings::[algorithm]_minimal_n`.

For some of the algorithms, there are even more tuning options, e. g. the ability to choose from multiple algorithm variants. See below for details.

### 5.363.3 Macro Definition Documentation

#### 5.363.3.1 `#define GLIBCXX_PARALLEL_CONDITION( __c )`

Determine at compile(?) -time if the parallel variant of an algorithm should be called.

##### Parameters

<code>__c</code>	A condition that is convertible to <code>bool</code> that is overruled by <code>__gnu_parallel::_Settings::algorithm_strategy</code> . Usually a decision based on the input size.
------------------	--

Definition at line 95 of file `settings.h`.

### 5.364 `shared_ptr.h` File Reference

#### Classes

- class `std::enable_shared_from_this< _Tp >`
- struct `std::hash< shared_ptr< _Tp > >`
- struct `std::owner_less< _Tp >`
- struct `std::owner_less< shared_ptr< _Tp > >`
- struct `std::owner_less< weak_ptr< _Tp > >`
- class `std::shared_ptr< _Tp >`
- class `std::weak_ptr< _Tp >`

#### Namespaces

- `std`

## Functions

- `template<typename _Tp, typename _Alloc, typename... _Args> shared_ptr< _Tp > std::allocate\_shared (const _Alloc &__a, _Args &&...__args)`
- `template<typename _Tp, typename _Tp1 > shared_ptr< _Tp > std::const\_pointer\_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Tp, typename _Tp1 > shared_ptr< _Tp > std::dynamic\_pointer\_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Del, typename _Tp, _Lock_policy _Lp> _Del * std::get\_deleter (const __shared_ptr< _Tp, _Lp > &__p) noexcept`
- `template<typename _Tp, typename... _Args> shared_ptr< _Tp > std::make\_shared (_Args &&...__args)`
- `template<typename _Tp1, typename _Tp2> bool std::operator!= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp > bool std::operator!= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp > bool std::operator!= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp1, typename _Tp2> bool std::operator< (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp > bool std::operator< (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp > bool std::operator< (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp> std::basic\_ostream< _Ch, _Tr > & std::operator<< (std::basic\_ostream< _Ch, _Tr > &__os, const __shared_ptr< _Tp, _Lp > &__p)`
- `template<typename _Tp1, typename _Tp2> bool std::operator<= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp > bool std::operator<= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp > bool std::operator<= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp1, typename _Tp2> bool std::operator== (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp > bool std::operator== (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp > bool std::operator== (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp1, typename _Tp2> bool std::operator> (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp > bool std::operator> (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp > bool std::operator> (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp1, typename _Tp2> bool std::operator>= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp > bool std::operator>= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp > bool std::operator>= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Tp1 > shared_ptr< _Tp > std::static\_pointer\_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Tp > void std::swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept`
- `template<typename _Tp > void std::swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept`

## 5.364.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 5.365 `shared_ptr_base.h` File Reference

### Classes

- class `std::bad_weak_ptr`
- class `std::enable_shared_from_this<_Tp>`
- struct `std::hash<__shared_ptr<_Tp, _Lp>>`
- struct `std::owner_less<_Tp>`
- class `std::shared_ptr<_Tp>`
- class `std::weak_ptr<_Tp>`

### Namespaces

- `std`

### Functions

- template<typename \_Tp, \_Lock\_policy \_Lp, typename \_Alloc, typename... \_Args> \_\_shared\_ptr<\_Tp, \_Lp> **std::allocate\_shared** (const \_Alloc &\_\_a, \_Args &&... \_\_args)
- template<\_Lock\_policy \_Lp, typename \_Tp1, typename \_Tp2> void **std::enable\_shared\_from\_this\_helper** (const \_\_shared\_count<\_Lp> &, const \_\_enable\_shared\_from\_this<\_Tp1, \_Lp> \*, const \_Tp2 \*) noexcept
- template<typename \_Tp1, typename \_Tp2> void **std::enable\_shared\_from\_this\_helper** (const \_\_shared\_count<> &, const enable\_shared\_from\_this<\_Tp1> \*, const \_Tp2 \*) noexcept
- template<\_Lock\_policy \_Lp> void **std::enable\_shared\_from\_this\_helper** (const \_\_shared\_count<\_Lp> &, ...) noexcept
- template<typename \_Tp, \_Lock\_policy \_Lp, typename... \_Args> \_\_shared\_ptr<\_Tp, \_Lp> **std::make\_shared** (\_Args &&... \_\_args)
- void **std::throw\_bad\_weak\_ptr** ()
- template<typename \_Tp, typename \_Tp1, \_Lock\_policy \_Lp> \_\_shared\_ptr<\_Tp, \_Lp> **std::const\_pointer\_cast** (const \_\_shared\_ptr<\_Tp1, \_Lp> &\_\_r) noexcept
- template<typename \_Tp, typename \_Tp1, \_Lock\_policy \_Lp> \_\_shared\_ptr<\_Tp, \_Lp> **std::dynamic\_pointer\_cast** (const \_\_shared\_ptr<\_Tp1, \_Lp> &\_\_r) noexcept
- template<typename \_Tp1, typename \_Tp2, \_Lock\_policy \_Lp> bool **std::operator!=** (const \_\_shared\_ptr<\_Tp1, \_Lp> &\_\_a, const \_\_shared\_ptr<\_Tp2, \_Lp> &\_\_b) noexcept
- template<typename \_Tp, \_Lock\_policy \_Lp> bool **std::operator!=** (const \_\_shared\_ptr<\_Tp, \_Lp> &\_\_a, nullptr\_t) noexcept
- template<typename \_Tp, \_Lock\_policy \_Lp> bool **std::operator!=** (nullptr\_t, const \_\_shared\_ptr<\_Tp, \_Lp> &\_\_a) noexcept
- template<typename \_Tp1, typename \_Tp2, \_Lock\_policy \_Lp> bool **std::operator<** (const \_\_shared\_ptr<\_Tp1, \_Lp> &\_\_a, const \_\_shared\_ptr<\_Tp2, \_Lp> &\_\_b) noexcept
- template<typename \_Tp, \_Lock\_policy \_Lp> bool **std::operator<** (const \_\_shared\_ptr<\_Tp, \_Lp> &\_\_a, nullptr\_t) noexcept
- template<typename \_Tp, \_Lock\_policy \_Lp> bool **std::operator<** (nullptr\_t, const \_\_shared\_ptr<\_Tp, \_Lp> &\_\_a) noexcept
- template<typename \_Tp1, typename \_Tp2, \_Lock\_policy \_Lp> bool **std::operator<=** (const \_\_shared\_ptr<\_Tp1, \_Lp> &\_\_a, const \_\_shared\_ptr<\_Tp2, \_Lp> &\_\_b) noexcept
- template<typename \_Tp, \_Lock\_policy \_Lp> bool **std::operator<=** (const \_\_shared\_ptr<\_Tp, \_Lp> &\_\_a, nullptr\_t) noexcept
- template<typename \_Tp, \_Lock\_policy \_Lp> bool **std::operator<=** (nullptr\_t, const \_\_shared\_ptr<\_Tp, \_Lp> &\_\_a) noexcept
- template<typename \_Tp1, typename \_Tp2, \_Lock\_policy \_Lp> bool **std::operator==** (const \_\_shared\_ptr<\_Tp1, \_Lp> &\_\_a, const \_\_shared\_ptr<\_Tp2, \_Lp> &\_\_b) noexcept

- `template<typename _Tp, _Lock_policy _Lp> bool std::operator== (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp> bool std::operator== (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp> bool std::operator> (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp> bool std::operator> (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp> bool std::operator> (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp> bool std::operator>= (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp> bool std::operator>= (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp> bool std::operator>= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp> __shared_ptr< _Tp, _Lp > std::static_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept`
- `template<typename _Tp, _Lock_policy _Lp> void std::swap (__shared_ptr< _Tp, _Lp > &__a, __shared_ptr< _Tp, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp> void std::swap (__weak_ptr< _Tp, _Lp > &__a, __weak_ptr< _Tp, _Lp > &__b) noexcept`

#### 5.365.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 5.366 size\_fn\_imps.hpp File Reference

#### 5.366.1 Detailed Description

Contains implementations of `cc_ht_map_'s` entire container size related functions.

## 5.367 slice\_array.h File Reference

#### Classes

- class [std::slice](#)
- class [std::slice\\_array< \\_Tp >](#)

#### Namespaces

- [std](#)

#### Macros

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

### 5.367.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 5.368 `sort.h` File Reference

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Functions

- `template<bool __stable, typename _RAIter, typename _Compare, typename _Parallelism> void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare> void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare> void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_exact_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare> void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_sampling_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare> void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, quicksort_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare> void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, balanced_quicksort_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare> void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, default_parallel_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare> void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, parallel_tag __parallelism)`

### 5.368.1 Detailed Description

Parallel sorting algorithm switch. This file is a GNU parallel extension to the Standard C++ Library.

## 5.369 `splay_fn_imps.hpp` File Reference

### 5.369.1 Detailed Description

Contains an implementation class for `splay_tree_`.

## 5.370 `splay_tree_.hpp` File Reference

### Classes

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

### Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- #define **PB\_DS\_ASSERT\_BASE\_NODE\_CONSISTENT**(\_Node)
- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_S\_TREE\_BASE**
- #define **PB\_DS\_S\_TREE\_BASE\_NAME**
- #define **PB\_DS\_S\_TREE\_NAME**

### 5.370.1 Detailed Description

Contains an implementation class for splay trees.

## 5.371 split\_fn\_imps.hpp File Reference

### 5.371.1 Detailed Description

Contains an implementation class for pat\_trie.

## 5.372 split\_join\_fn\_imps.hpp File Reference

### 5.372.1 Detailed Description

Contains an implementation class for a binary\_heap.

## 5.373 split\_join\_fn\_imps.hpp File Reference

### 5.373.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

## 5.374 split\_join\_fn\_imps.hpp File Reference

### 5.374.1 Detailed Description

Contains an implementation class for bin\_search\_tree\_.

## 5.375 split\_join\_fn\_imps.hpp File Reference

### 5.375.1 Detailed Description

Contains an implementation class for ov\_tree\_.

## 5.376 split\_join\_fn\_imps.hpp File Reference

### 5.376.1 Detailed Description

Contains an implementation class for a pairing heap.

**5.377 split\_join\_fn\_imps.hpp File Reference****5.377.1 Detailed Description**

Contains an implementation for rb\_tree\_.

**5.378 split\_join\_fn\_imps.hpp File Reference****5.378.1 Detailed Description**

Contains an implementation for rc\_binomial\_heap\_.

**5.379 split\_join\_fn\_imps.hpp File Reference****5.379.1 Detailed Description**

Contains an implementation class for splay\_tree\_.

**5.380 split\_join\_fn\_imps.hpp File Reference****5.380.1 Detailed Description**

Contains an implementation for thin\_heap\_.

**5.381 sso\_string\_base.h File Reference****Namespaces**

- [\\_\\_gnu\\_cxx](#)

**5.381.1 Detailed Description**

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

**5.382 standard\_policies.hpp File Reference****Classes**

- [struct \\_\\_gnu\\_pbds::detail::default\\_comb\\_hash\\_fn](#)
- [struct \\_\\_gnu\\_pbds::detail::default\\_eq\\_fn< Key >](#)
- [struct \\_\\_gnu\\_pbds::detail::default\\_hash\\_fn< Key >](#)
- [struct \\_\\_gnu\\_pbds::detail::default\\_probe\\_fn< Comb\\_Probe\\_Fn >](#)
- [struct \\_\\_gnu\\_pbds::detail::default\\_resize\\_policy< Comb\\_Hash\\_Fn >](#)
- [struct \\_\\_gnu\\_pbds::detail::default\\_trie\\_access\\_traits< Key >](#)
- [struct \\_\\_gnu\\_pbds::detail::default\\_trie\\_access\\_traits< std::basic\\_string< Char, Char\\_Traits, std::allocator< char > > >](#)
- [struct \\_\\_gnu\\_pbds::detail::default\\_update\\_policy](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define __dtrie_alloc`
- `#define __dtrie_string`

## Enumerations

- enum { **default\_store\_hash** }

## 5.382.1 Detailed Description

Contains standard policies for containers.

**5.383 stdc++.h File Reference**

## 5.383.1 Detailed Description

This is an implementation file for a precompiled header.

**5.384 stdio\_filebuf.h File Reference**

## Classes

- class [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits>](#)

## Namespaces

- [\\_\\_gnu\\_cxx](#)

## 5.384.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

**5.385 stdio\_sync\_filebuf.h File Reference**

## Classes

- class [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf<\\_CharT, \\_Traits>](#)

## Namespaces

- [\\_\\_gnu\\_cxx](#)



### 5.385.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.386 stdtr1c++.h File Reference

### 5.386.1 Detailed Description

This is an implementation file for a precompiled header.

## 5.387 stl\_algo.h File Reference

### Namespaces

- [std](#)

### Enumerations

- enum { **\_S\_threshold** }
- enum { **\_S\_chunk\_size** }

### Functions

- template<typename \_RandomAccessIterator, typename \_Distance> void **std::\_\_chunk\_insertion\_sort** (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last, \_Distance \_\_chunk\_size)
- template<typename \_RandomAccessIterator, typename \_Distance, typename \_Compare> void **std::\_\_chunk\_insertion\_sort** (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last, \_Distance \_\_chunk\_size, \_Compare \_\_comp)
- template<typename \_InputIterator, typename \_Size, typename \_OutputIterator> \_OutputIterator **std::\_\_copy\_n** (\_InputIterator \_\_first, \_Size \_\_n, \_OutputIterator \_\_result, input\_iterator\_tag)
- template<typename \_RandomAccessIterator, typename \_Size, typename \_OutputIterator> \_OutputIterator **std::\_\_copy\_n** (\_RandomAccessIterator \_\_first, \_Size \_\_n, \_OutputIterator \_\_result, random\_access\_iterator\_tag)
- template<typename \_RandomAccessIterator> void **std::\_\_final\_insertion\_sort** (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last)
- template<typename \_RandomAccessIterator, typename \_Compare> void **std::\_\_final\_insertion\_sort** (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last, \_Compare \_\_comp)
- template<typename \_InputIterator, typename \_Tp> \_InputIterator **std::\_\_find** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Tp &\_\_val, input\_iterator\_tag)
- template<typename \_RandomAccessIterator, typename \_Tp> \_RandomAccessIterator **std::\_\_find** (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last, const \_Tp &\_\_val, random\_access\_iterator\_tag)
- template<typename \_ForwardIterator1, typename \_ForwardIterator2> \_ForwardIterator1 **std::\_\_find\_end** (\_ForwardIterator1 \_\_first1, \_ForwardIterator1 \_\_last1, \_ForwardIterator2 \_\_first2, \_ForwardIterator2 \_\_last2, forward\_iterator\_tag, forward\_iterator\_tag)
- template<typename \_ForwardIterator1, typename \_ForwardIterator2, typename \_BinaryPredicate> \_ForwardIterator1 **std::\_\_find\_end** (\_ForwardIterator1 \_\_first1, \_ForwardIterator1 \_\_last1, \_ForwardIterator2 \_\_first2, \_ForwardIterator2 \_\_last2, forward\_iterator\_tag, forward\_iterator\_tag, \_BinaryPredicate \_\_comp)
- template<typename \_BidirectionalIterator1, typename \_BidirectionalIterator2> \_BidirectionalIterator1 **std::\_\_find\_end** (\_BidirectionalIterator1 \_\_first1, \_BidirectionalIterator1 \_\_last1, \_BidirectionalIterator2 \_\_first2, \_BidirectionalIterator2 \_\_last2, bidirectional\_iterator\_tag, bidirectional\_iterator\_tag)
- template<typename \_BidirectionalIterator1, typename \_BidirectionalIterator2, typename \_BinaryPredicate> \_BidirectionalIterator1 **std::\_\_find\_end** (\_BidirectionalIterator1 \_\_first1, \_BidirectionalIterator1 \_\_last1, \_BidirectionalIterator2 \_\_first2, \_BidirectionalIterator2 \_\_last2, bidirectional\_iterator\_tag, bidirectional\_iterator\_tag, \_BinaryPredicate \_\_comp)

- `template<typename _InputIterator, typename _Predicate> _InputIterator std::\_\_find\_if (_InputIterator __first, _InputIterator __last, _Predicate __pred, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Predicate> _RandomAccessIterator std::\_\_find\_if (_RandomAccessIterator __first, _RandomAccessIterator __last, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _InputIterator, typename _Predicate> _InputIterator std::\_\_find\_if\_not (_InputIterator __first, _InputIterator __last, _Predicate __pred, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Predicate> _RandomAccessIterator std::\_\_find\_if\_not (_RandomAccessIterator __first, _RandomAccessIterator __last, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _InputIterator, typename _Predicate> _InputIterator std::\_\_find\_if\_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate, typename _Distance> _InputIterator std::\_\_find\_if\_not\_n (_InputIterator __first, _Distance &__len, _Predicate __pred)`
- `template<typename _EuclideanRingElement> _EuclideanRingElement std::\_\_gcd (_EuclideanRingElement __m, _EuclideanRingElement __n)`
- `template<typename _RandomAccessIterator> void std::\_\_heap\_select (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare> void std::\_\_heap\_select (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Distance> _ForwardIterator std::\_\_inplace\_stable\_partition (_ForwardIterator __first, _Predicate __pred, _Distance __len)`
- `template<typename _RandomAccessIterator> void std::\_\_inplace\_stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare> void std::\_\_inplace\_stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator> void std::\_\_insertion\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare> void std::\_\_insertion\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Size> void std::\_\_introsselect (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Size __depth_limit)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare> void std::\_\_introsselect (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Size> void std::\_\_introsort\_loop (_RandomAccessIterator __first, _RandomAccessIterator __last, _Size __depth_limit)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare> void std::\_\_introsort\_loop (_RandomAccessIterator __first, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer> void std::\_\_merge\_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _Compare> void std::\_\_merge\_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)`
- `template<typename _RandomAccessIterator1, typename _RandomAccessIterator2, typename _Distance> void std::\_\_merge\_sort\_loop (_RandomAccessIterator1 __first, _RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance __step_size)`
- `template<typename _RandomAccessIterator1, typename _RandomAccessIterator2, typename _Distance, typename _Compare> void std::\_\_merge\_sort\_loop (_RandomAccessIterator1 __first, _RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance __step_size, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Pointer> void std::\_\_merge\_sort\_with\_buffer (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Compare> void std::\_\_merge\_sort\_with\_buffer (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Compare __comp)`

- `template<typename _BidirectionalIterator, typename _Distance> void std::__merge_without_buffer (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Compare> void std::__merge_without_buffer (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Compare __comp)`
- `template<typename _Iterator> void std::__move_median_to_first (_Iterator __result, _Iterator __a, _Iterator __b, _Iterator __c)`
- `template<typename _Iterator, typename _Compare> void std::__move_median_to_first (_Iterator __result, _Iterator __a, _Iterator __b, _Iterator __c, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator> _OutputIterator std::__move_merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare> _OutputIterator std::__move_merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator> void std::__move_merge_adaptive (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare> void std::__move_merge_adaptive (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BidirectionalIterator3> void std::__move_merge_adaptive_backward (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BidirectionalIterator3, typename _Compare> void std::__move_merge_adaptive_backward (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate> _ForwardIterator std::__partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, forward_iterator_tag)`
- `template<typename _BidirectionalIterator, typename _Predicate> _BidirectionalIterator std::__partition (_BidirectionalIterator __first, _BidirectionalIterator __last, _Predicate __pred, bidirectional_iterator_tag)`
- `template<typename _BidirectionalIterator> void std::__reverse (_BidirectionalIterator __first, _BidirectionalIterator __last, bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator> void std::__reverse (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _ForwardIterator> void std::__rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, forward_iterator_tag)`
- `template<typename _BidirectionalIterator> void std::__rotate (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator> void std::__rotate (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _Distance> _BidirectionalIterator1 std::__rotate_adaptive (_BidirectionalIterator1 __first, _BidirectionalIterator1 __middle, _BidirectionalIterator1 __last, _Distance __len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance __buffer_size)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp> _ForwardIterator std::__search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, std::forward_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Integer, typename _Tp> _RandomAccessIterator std::__search_n (_RandomAccessIterator __first, _RandomAccessIterator __last, _Integer __count, const _Tp &__val, std::random_access_iterator_tag)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate> _ForwardIterator std::__search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, std::forward_iterator_tag)`

- `template<typename _RandomAccessIter, typename _Integer, typename _Tp, typename _BinaryPredicate> _RandomAccessIter std::\_\_search\_n (_RandomAccessIter __first, _RandomAccessIter __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, std::random\_access\_iterator\_tag)`
- `template<typename _ForwardIterator, typename _Pointer, typename _Predicate, typename _Distance> _ForwardIterator std::\_\_stable\_partition\_adaptive (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, _Distance __len, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Distance> void std::\_\_stable\_sort\_adaptive (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Distance, typename _Compare> void std::\_\_stable\_sort\_adaptive (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)`
- `template<typename _RandomAccessIterator> void std::\_\_unguarded\_insertion\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare> void std::\_\_unguarded\_insertion\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator> void std::\_\_unguarded\_linear\_insert (_RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare> void std::\_\_unguarded\_linear\_insert (_RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Tp> _RandomAccessIterator std::\_\_unguarded\_partition (_RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp &__pivot)`
- `template<typename _RandomAccessIterator, typename _Tp, typename _Compare> _RandomAccessIterator std::\_\_unguarded\_partition (_RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp &__pivot, _Compare __comp)`
- `template<typename _RandomAccessIterator> _RandomAccessIterator std::\_\_unguarded\_partition\_pivot (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare> _RandomAccessIterator std::\_\_unguarded\_partition\_pivot (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _OutputIterator> _OutputIterator std::\_\_unique\_copy (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __result, forward\_iterator\_tag, output\_iterator\_tag)`
- `template<typename _InputIterator, typename _OutputIterator> _OutputIterator std::\_\_unique\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, input\_iterator\_tag, output\_iterator\_tag)`
- `template<typename _InputIterator, typename _ForwardIterator> _ForwardIterator std::\_\_unique\_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, input\_iterator\_tag, forward\_iterator\_tag)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _BinaryPredicate> _OutputIterator std::\_\_unique\_copy (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, forward\_iterator\_tag, output\_iterator\_tag)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate> _OutputIterator std::\_\_unique\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, input\_iterator\_tag, output\_iterator\_tag)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate> _ForwardIterator std::\_\_unique\_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _BinaryPredicate __binary_pred, input\_iterator\_tag, forward\_iterator\_tag)`
- `template<typename _ForwardIterator> _ForwardIterator std::adjacent\_find (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate> _ForwardIterator std::adjacent\_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Predicate> bool std::all\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate> bool std::any\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp> bool std::binary\_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`

- `template<typename _ForwardIterator, typename _Tp, typename _Compare> bool std::binary\_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate> _OutputIterator std::copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator> _OutputIterator std::copy\_n (_InputIterator __first, _Size __n, _OutputIterator __result)`
- `template<typename _InputIterator, typename _Tp> iterator_traits< _InputIterator >::difference_type std::count (_InputIterator __first, _InputIterator __last, const _Tp &__value)`
- `template<typename _InputIterator, typename _Predicate> iterator_traits< _InputIterator >::difference_type std::count\_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp> pair< _ForwardIterator, _ForwardIterator > std::equal\_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare> pair< _ForwardIterator, _ForwardIterator > std::equal\_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _InputIterator, typename _Tp> _InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2> _ForwardIterator1 std::find\_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate> _ForwardIterator1 std::find\_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _ForwardIterator> _InputIterator std::find\_first\_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate> _InputIterator std::find\_first\_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _Predicate> _InputIterator std::find\_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate> _InputIterator std::find\_if\_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Function> _Function std::for\_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `template<typename _ForwardIterator, typename _Generator> void std::generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator> _OutputIterator std::generate\_n (_OutputIterator __first, _Size __n, _Generator __gen)`
- `template<typename _InputIterator1, typename _InputIterator2> bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare> bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _BidirectionalIterator> void std::inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare> void std::inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _Predicate> bool std::is\_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2> bool std::is\_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate> bool std::is\_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)`
- `template<typename _ForwardIterator> bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare> bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`

- `template<typename _ForwardIterator> _ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare> _ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare> _ForwardIterator std::lower\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, _Compare __comp)`
- `template<typename _Tp> _Tp std::max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare> _Tp std::max (initializer_list< _Tp >, _Compare)`
- `template<typename _ForwardIterator> _ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare> _ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator> _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare> _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Tp> _Tp std::min (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare> _Tp std::min (initializer_list< _Tp >, _Compare)`
- `template<typename _ForwardIterator> _ForwardIterator std::min\_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare> _ForwardIterator std::min\_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Tp> pair< const _Tp &, const _Tp &> std::minmax (const _Tp & __a, const _Tp & __b)`
- `template<typename _Tp, typename _Compare> pair< const _Tp &, const _Tp &> std::minmax (const _Tp & __a, const _Tp & __b, _Compare __comp)`
- `template<typename _Tp> pair< _Tp, _Tp> std::minmax (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare> pair< _Tp, _Tp> std::minmax (initializer_list< _Tp >, _Compare)`
- `template<typename _ForwardIterator> pair< _ForwardIterator, _ForwardIterator> std::minmax\_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare> pair< _ForwardIterator, _ForwardIterator> std::minmax\_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _BidirectionalIterator> bool std::next\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare> bool std::next\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _Predicate> bool std::none\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _RandomAccessIterator> void std::nth\_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare> void std::nth\_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator> void std::partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare> void std::partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator> _RandomAccessIterator std::partial\_sort\_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare> _RandomAccessIterator std::partial\_sort\_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)`



- `template<typename _ForwardIterator, typename _Predicate> _ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate> pair<_OutputIterator1, _OutputIterator2> std::partition\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate> _ForwardIterator std::partition\_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _BidirectionalIterator> bool std::prev\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare> bool std::prev\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator> void std::random\_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator> void std::random\_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator && __rand)`
- `template<typename _ForwardIterator, typename _Tp> _ForwardIterator std::remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp> _OutputIterator std::remove\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp & __value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate> _OutputIterator std::remove\_copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate> _ForwardIterator std::remove\_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp> void std::replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __old_value, const _Tp & __new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp> _OutputIterator std::replace\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp & __old_value, const _Tp & __new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp> _OutputIterator std::replace\_copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp & __new_value)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Tp> void std::replace\_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp & __new_value)`
- `template<typename _BidirectionalIterator> void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _OutputIterator> _OutputIterator std::reverse\_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result)`
- `template<typename _ForwardIterator> void std::rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _OutputIterator> _OutputIterator std::rotate\_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, _OutputIterator __result)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2> _ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate> _ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp> _ForwardIterator std::search\_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp & __val)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate> _ForwardIterator std::search\_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp & __val, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator> _OutputIterator std::set\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`

- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare> _OutputIterator std::set\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator> _OutputIterator std::set\_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare> _OutputIterator std::set\_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator> _OutputIterator std::set\_symmetric\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare> _OutputIterator std::set\_symmetric\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator> _OutputIterator std::set\_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare> _OutputIterator std::set\_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator> void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumberGenerator && __g)`
- `template<typename _RandomAccessIterator> void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare> void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate> _ForwardIterator std::stable\_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _RandomAccessIterator> void std::stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare> void std::stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation> _OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation> _OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _ForwardIterator> _ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate> _ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _OutputIterator> _OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate> _OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator, typename _Tp> _ForwardIterator std::upper\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare> _ForwardIterator std::upper\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, _Compare __comp)`



### 5.387.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

## 5.388 `stl_algobase.h` File Reference

### Classes

- struct [std::char\\_traits<\\_CharT>](#)
- class [std::istreambuf\\_iterator<\\_CharT, \\_Traits>](#)
- class [std::ostreambuf\\_iterator<\\_CharT, \\_Traits>](#)

### Namespaces

- [std](#)

### Macros

- `#define GLIBCXX_MOVE3(_Tp, _Up, _Vp)`
- `#define GLIBCXX_MOVE_BACKWARD3(_Tp, _Up, _Vp)`

### Functions

- `template<bool _IsMove, typename _II, typename _OI> _OI std::__copy_move_a(_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _CharT> __gnu_cxx::__enable_if<__is_char<_CharT>::__value, ostreambuf_iterator<_CharT, char_traits<_CharT>>>::__type std::__copy_move_a2(_CharT *, _CharT *, ostreambuf_iterator<_CharT, char_traits<_CharT>>)`
- `template<bool _IsMove, typename _CharT> __gnu_cxx::__enable_if<__is_char<_CharT>::__value, ostreambuf_iterator<_CharT, char_traits<_CharT>>>::__type std::__copy_move_a2(const _CharT *, const _CharT *, ostreambuf_iterator<_CharT, char_traits<_CharT>>)`
- `template<bool _IsMove, typename _CharT> __gnu_cxx::__enable_if<__is_char<_CharT>::__value, _CharT *>::__type std::__copy_move_a2(istreambuf_iterator<_CharT, char_traits<_CharT>>, istreambuf_iterator<_CharT, char_traits<_CharT>>, _CharT *)`
- `template<bool _IsMove, typename _II, typename _OI> _OI std::__copy_move_a2(_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2> _BI2 std::__copy_move_backward_a(_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2> _BI2 std::__copy_move_backward_a2(_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _II1, typename _II2> bool std::__equal_aux(_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _ForwardIterator, typename _Tp> __gnu_cxx::__enable_if<!__is_scalar<_Tp>::__value, void>::__type std::__fill_a(_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _ForwardIterator, typename _Tp> __gnu_cxx::__enable_if<__is_scalar<_Tp>::__value, void>::__type std::__fill_a(_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _Tp> __gnu_cxx::__enable_if<__is_byte<_Tp>::__value, void>::__type std::__fill_a(_Tp * __first, _Tp * __last, const _Tp &__c)`
- `template<typename _OutputIterator, typename _Size, typename _Tp> __gnu_cxx::__enable_if<!__is_scalar<_Tp>::__value, _OutputIterator>::__type std::__fill_n_a(_OutputIterator __first, _Size __n, const _Tp &__value)`
- `template<typename _OutputIterator, typename _Size, typename _Tp> __gnu_cxx::__enable_if<__is_scalar<_Tp>::__value, _OutputIterator>::__type std::__fill_n_a(_OutputIterator __first, _Size __n, const _Tp &__value)`

- `template<typename _Size, typename _Tp> __gnu_cxx::__enable_if<__is_byte<_Tp>::__value, _Tp * >::__type std::__fill_n_a(_Tp * __first, _Size __n, const _Tp & __c)`
- `template<typename _II1, typename _II2> bool std::__lexicographical_compare_aux(_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `constexpr int std::__lg(int __n)`
- `constexpr unsigned std::__lg(unsigned __n)`
- `constexpr long std::__lg(long __n)`
- `constexpr unsigned long std::__lg(unsigned long __n)`
- `constexpr long long std::__lg(long long __n)`
- `constexpr unsigned long long std::__lg(unsigned long long __n)`
- `template<typename _Iterator> _Miter_base<_Iterator>::iterator_type std::__miter_base(_Iterator __it)`
- `template<typename _Iterator> _Niter_base<_Iterator>::iterator_type std::__niter_base(_Iterator __it)`
- `template<typename _II, typename _OI> _OI std::copy(_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2> _BI2 std::copy_backward(_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _IIter1, typename _IIter2, typename _BinaryPredicate> bool std::equal(_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _II1, typename _II2> bool std::equal(_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _ForwardIterator, typename _Tp> void std::fill(_ForwardIterator __first, _ForwardIterator __last, const _Tp & __value)`
- `template<typename _OI, typename _Size, typename _Tp> _OI std::fill_n(_OI __first, _Size __n, const _Tp & __value)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2> void std::iter_swap(_ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _II1, typename _II2> bool std::lexicographical_compare(_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _II1, typename _II2, typename _Compare> bool std::lexicographical_compare(_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp> _ForwardIterator std::lower_bound(_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val)`
- `template<typename _Tp> const _Tp & std::max(const _Tp & __a, const _Tp & __b)`
- `template<typename _Tp, typename _Compare> const _Tp & std::max(const _Tp & __a, const _Tp & __b, _Compare __comp)`
- `template<typename _Tp> const _Tp & std::min(const _Tp & __a, const _Tp & __b)`
- `template<typename _Tp, typename _Compare> const _Tp & std::min(const _Tp & __a, const _Tp & __b, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2> pair<_InputIterator1, _InputIterator2> std::mismatch(_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate> pair<_InputIterator1, _InputIterator2> std::mismatch(_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _II, typename _OI> _OI std::move(_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2> _BI2 std::move_backward(_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2> _ForwardIterator2 std::swap_ranges(_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`

### 5.388.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

### 5.389 `std::bvector.h` File Reference

#### Classes

- struct `std::hash<::vector< bool, _Alloc > >`
- class `std::vector< bool, _Alloc >`

#### Namespaces

- `std`

#### Typedefs

- typedef unsigned long `std::_Bit_type`

#### Enumerations

- enum { `_S_word_bit` }

#### Functions

- void `std::_fill_bvector` (`_Bit_iterator __first`, `_Bit_iterator __last`, `bool __x`)
- void `std::fill` (`_Bit_iterator __first`, `_Bit_iterator __last`, `const bool &__x`)
- `_Bit_iterator std::operator+` (`ptrdiff_t __n`, `const _Bit_iterator &__x`)
- `_Bit_const_iterator std::operator+` (`ptrdiff_t __n`, `const _Bit_const_iterator &__x`)
- `ptrdiff_t std::operator-` (`const _Bit_iterator_base &__x`, `const _Bit_iterator_base &__y`)
- void `std::swap` (`_Bit_reference __x`, `_Bit_reference __y`) `noexcept`
- void `std::swap` (`_Bit_reference __x`, `bool &__y`) `noexcept`
- void `std::swap` (`bool &__x`, `_Bit_reference __y`) `noexcept`

#### 5.389.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<vector>`.

### 5.390 `std::construct.h` File Reference

#### Namespaces

- `std`

#### Functions

- template<typename `_T1`, typename... `_Args`> void `std::_Construct` (`_T1 *__p`, `_Args &&...__args`)
- template<typename `_Tp`> void `std::_Destroy` (`_Tp *__pointer`)
- template<typename `_ForwardIterator`> void `std::_Destroy` (`_ForwardIterator __first`, `_ForwardIterator __last`)
- template<typename `_ForwardIterator`, typename `_Allocator`> void `std::_Destroy` (`_ForwardIterator __first`, `_ForwardIterator __last`, `_Allocator &__alloc`)
- template<typename `_ForwardIterator`, typename `_Tp`> void `std::_Destroy` (`_ForwardIterator __first`, `_ForwardIterator __last`, `allocator<_Tp> &`)

## 5.390.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

5.391 `std_deque.h` File Reference

## Classes

- class `std::_Deque_base<_Tp, _Alloc>`
- struct `std::_Deque_iterator<_Tp, _Ref, _Ptr>`
- class `std::deque<_Tp, _Alloc>`

## Namespaces

- `std`

## Macros

- `#define _GLIBCXX_DEQUE_BUF_SIZE`

## Functions

- `size_t std::__deque_buf_size (size_t __size)`
- `template<typename _Tp> _Deque_iterator<_Tp, _Tp&, _Tp*> std::copy (_Deque_iterator<_Tp, const _Tp&, const _Tp*>, const _Tp*>, _Deque_iterator<_Tp, const _Tp&, const _Tp*>, _Deque_iterator<_Tp, _Tp&, _Tp*>)`
- `template<typename _Tp> _Deque_iterator<_Tp, _Tp&, _Tp*> std::copy (_Deque_iterator<_Tp, _Tp&, _Tp*> __first, _Deque_iterator<_Tp, _Tp&, _Tp*> __last, _Deque_iterator<_Tp, _Tp&, _Tp*> __result)`
- `template<typename _Tp> _Deque_iterator<_Tp, _Tp&, _Tp*> std::copy_backward (_Deque_iterator<_Tp, const _Tp&, const _Tp*>, _Deque_iterator<_Tp, const _Tp&, const _Tp*>, _Deque_iterator<_Tp, _Tp&, _Tp*>)`
- `template<typename _Tp> _Deque_iterator<_Tp, _Tp&, _Tp*> std::copy_backward (_Deque_iterator<_Tp, _Tp&, _Tp*> __first, _Deque_iterator<_Tp, _Tp&, _Tp*> __last, _Deque_iterator<_Tp, _Tp&, _Tp*> __result)`
- `template<typename _Tp> void std::fill (const _Deque_iterator<_Tp, _Tp&, _Tp*> &, const _Deque_iterator<_Tp, _Tp&, _Tp*> &, const _Tp&)`
- `template<typename _Tp> _Deque_iterator<_Tp, _Tp&, _Tp*> std::move (_Deque_iterator<_Tp, const _Tp&, const _Tp*>, _Deque_iterator<_Tp, const _Tp&, const _Tp*>, _Deque_iterator<_Tp, _Tp&, _Tp*>)`
- `template<typename _Tp> _Deque_iterator<_Tp, _Tp&, _Tp*> std::move (_Deque_iterator<_Tp, _Tp&, _Tp*> __first, _Deque_iterator<_Tp, _Tp&, _Tp*> __last, _Deque_iterator<_Tp, _Tp&, _Tp*> __result)`
- `template<typename _Tp> _Deque_iterator<_Tp, _Tp&, _Tp*> std::move_backward (_Deque_iterator<_Tp, const _Tp&, const _Tp*>, _Deque_iterator<_Tp, const _Tp&, const _Tp*>, _Deque_iterator<_Tp, _Tp&, _Tp*>)`
- `template<typename _Tp> _Deque_iterator<_Tp, _Tp&, _Tp*> std::move_backward (_Deque_iterator<_Tp, _Tp&, _Tp*> __first, _Deque_iterator<_Tp, _Tp&, _Tp*> __last, _Deque_iterator<_Tp, _Tp&, _Tp*> __result)`
- `template<typename _Tp, typename _Ref, typename _Ptr> bool std::operator!= (const _Deque_iterator<_Tp, _Ref, _Ptr> &__x, const _Deque_iterator<_Tp, _Ref, _Ptr> &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR> bool std::operator!= (const _Deque_iterator<_Tp, _RefL, _PtrL> &__x, const _Deque_iterator<_Tp, _RefR, _PtrR> &__y)`

- `template<typename _Tp, typename _Alloc> bool std::operator!= (const deque< _Tp, _Alloc> &__x, const deque< _Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr> _Deque_iterator< _Tp, _Ref, _Ptr> std::operator+ (ptrdiff_t __n, const _Deque_iterator< _Tp, _Ref, _Ptr> &__x)`
- `template<typename _Tp, typename _Ref, typename _Ptr> _Deque_iterator< _Tp, _Ref, _Ptr> ::difference_type std::operator- (const _Deque_iterator< _Tp, _Ref, _Ptr> &__x, const _Deque_iterator< _Tp, _Ref, _Ptr> &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR> _Deque_iterator< _Tp, _RefL, _PtrL> ::difference_type std::operator- (const _Deque_iterator< _Tp, _RefL, _PtrL> &__x, const _Deque_iterator< _Tp, _RefR, _PtrR> &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr> bool std::operator< (const _Deque_iterator< _Tp, _Ref, _Ptr> &__x, const _Deque_iterator< _Tp, _Ref, _Ptr> &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR> bool std::operator< (const _Deque_iterator< _Tp, _RefL, _PtrL> &__x, const _Deque_iterator< _Tp, _RefR, _PtrR> &__y)`
- `template<typename _Tp, typename _Alloc> bool std::operator< (const deque< _Tp, _Alloc> &__x, const deque< _Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr> bool std::operator<= (const _Deque_iterator< _Tp, _Ref, _Ptr> &__x, const _Deque_iterator< _Tp, _Ref, _Ptr> &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR> bool std::operator<= (const _Deque_iterator< _Tp, _RefL, _PtrL> &__x, const _Deque_iterator< _Tp, _RefR, _PtrR> &__y)`
- `template<typename _Tp, typename _Alloc> bool std::operator<= (const deque< _Tp, _Alloc> &__x, const deque< _Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr> bool std::operator== (const _Deque_iterator< _Tp, _Ref, _Ptr> &__x, const _Deque_iterator< _Tp, _Ref, _Ptr> &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR> bool std::operator== (const _Deque_iterator< _Tp, _RefL, _PtrL> &__x, const _Deque_iterator< _Tp, _RefR, _PtrR> &__y)`
- `template<typename _Tp, typename _Alloc> bool std::operator== (const deque< _Tp, _Alloc> &__x, const deque< _Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr> bool std::operator> (const _Deque_iterator< _Tp, _Ref, _Ptr> &__x, const _Deque_iterator< _Tp, _Ref, _Ptr> &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR> bool std::operator> (const _Deque_iterator< _Tp, _RefL, _PtrL> &__x, const _Deque_iterator< _Tp, _RefR, _PtrR> &__y)`
- `template<typename _Tp, typename _Alloc> bool std::operator> (const deque< _Tp, _Alloc> &__x, const deque< _Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr> bool std::operator>= (const _Deque_iterator< _Tp, _Ref, _Ptr> &__x, const _Deque_iterator< _Tp, _Ref, _Ptr> &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR> bool std::operator>= (const _Deque_iterator< _Tp, _RefL, _PtrL> &__x, const _Deque_iterator< _Tp, _RefR, _PtrR> &__y)`
- `template<typename _Tp, typename _Alloc> bool std::operator>= (const deque< _Tp, _Alloc> &__x, const deque< _Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc> void std::swap (deque< _Tp, _Alloc> &__x, deque< _Tp, _Alloc> &__y)`

### 5.391.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<deque>`.

### 5.391.2 Macro Definition Documentation

#### 5.391.2.1 `#define _GLIBCXX_DEQUE_BUF_SIZE`

This function controls the size of memory nodes.

## Parameters

<code>__size</code>	The size of an element.
---------------------	-------------------------

## Returns

The number (not byte size) of elements per node.

This function started off as a compiler kludge from SGI, but seems to be a useful wrapper around a repeated constant expression. The **512** is tunable (and no other code needs to change), but no investigation has been done since inheriting the SGI code. Touch `_GLIBCXX_DEQUE_BUF_SIZE` only if you know what you are doing, however: changing it breaks the binary compatibility!!

Definition at line 85 of file `std_deque.h`.

5.392 `std_function.h` File Reference

## Classes

- struct `std::binary_function<_Arg1, _Arg2, _Result >`
- class `std::binary_negate<_Predicate >`
- class `std::const_mem_fun1_ref_t<_Ret, _Tp, _Arg >`
- class `std::const_mem_fun1_t<_Ret, _Tp, _Arg >`
- class `std::const_mem_fun_ref_t<_Ret, _Tp >`
- class `std::const_mem_fun_t<_Ret, _Tp >`
- struct `std::divides<_Tp >`
- struct `std::equal_to<_Tp >`
- struct `std::greater<_Tp >`
- struct `std::greater_equal<_Tp >`
- struct `std::less<_Tp >`
- struct `std::less_equal<_Tp >`
- struct `std::logical_and<_Tp >`
- struct `std::logical_not<_Tp >`
- struct `std::logical_or<_Tp >`
- class `std::mem_fun1_ref_t<_Ret, _Tp, _Arg >`
- class `std::mem_fun1_t<_Ret, _Tp, _Arg >`
- class `std::mem_fun_ref_t<_Ret, _Tp >`
- class `std::mem_fun_t<_Ret, _Tp >`
- struct `std::minus<_Tp >`
- struct `std::modulus<_Tp >`
- struct `std::multiplies<_Tp >`
- struct `std::negate<_Tp >`
- struct `std::not_equal_to<_Tp >`
- struct `std::plus<_Tp >`
- class `std::pointer_to_binary_function<_Arg1, _Arg2, _Result >`
- class `std::pointer_to_unary_function<_Arg, _Result >`
- struct `std::unary_function<_Arg, _Result >`
- class `std::unary_negate<_Predicate >`

## Namespaces

- `std`

## Functions

- `template<typename _Ret, typename _Tp> mem_fun_t< _Ret, _Tp> std::mem_fun (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp> const_mem_fun_t< _Ret, _Tp> std::mem_fun (_Ret(_Tp::*__f)() const)`
- `template<typename _Ret, typename _Tp, typename _Arg> mem_fun1_t< _Ret, _Tp, _Arg> std::mem_fun (_Ret(_Tp::*__f)(_Arg))`
- `template<typename _Ret, typename _Tp, typename _Arg> const_mem_fun1_t< _Ret, _Tp, _Arg> std::mem_fun (_Ret(_Tp::*__f)(_Arg) const)`
- `template<typename _Ret, typename _Tp> mem_fun_ref_t< _Ret, _Tp> std::mem_fun_ref (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp> const_mem_fun_ref_t< _Ret, _Tp> std::mem_fun_ref (_Ret(_Tp::*__f)() const)`
- `template<typename _Ret, typename _Tp, typename _Arg> mem_fun1_ref_t< _Ret, _Tp, _Arg> std::mem_fun_ref (_Ret(_Tp::*__f)(_Arg))`
- `template<typename _Ret, typename _Tp, typename _Arg> const_mem_fun1_ref_t< _Ret, _Tp, _Arg> std::mem_fun_ref (_Ret(_Tp::*__f)(_Arg) const)`
- `template<typename _Predicate> unary_negate< _Predicate> std::not1 (const _Predicate &__pred)`
- `template<typename _Predicate> binary_negate< _Predicate> std::not2 (const _Predicate &__pred)`
- `template<typename _Arg, typename _Result> pointer_to_unary_function< _Arg, _Result> std::ptr_fun (_Result(*__x)(_Arg))`
- `template<typename _Arg1, typename _Arg2, typename _Result> pointer_to_binary_function< _Arg1, _Arg2, _Result> std::ptr_fun (_Result(*__x)(_Arg1, _Arg2))`

### 5.392.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 5.393 `stl_heap.h` File Reference

### Namespaces

- [std](#)

### Functions

- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp> void std::__adjust_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __len, _Tp __value)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare> void std::__adjust_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __len, _Tp __value, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance> bool std::__is_heap (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator, typename _Compare, typename _Distance> bool std::__is_heap (_RandomAccessIterator __first, _Compare __comp, _Distance __n)`
- `template<typename _RandomAccessIterator> bool std::__is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare> bool std::__is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance> _Distance std::__is_heap_until (_RandomAccessIterator __first, _Distance __n)`

- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare> _Distance std::__is_heap_until (↵  
_RandomAccessIterator __first, _Distance __n, _Compare __comp)`
- `template<typename _RandomAccessIterator> void std::__pop_heap (_RandomAccessIterator __first, _Random↵  
AccessIterator __last, _RandomAccessIterator __result)`
- `template<typename _RandomAccessIterator, typename _Compare> void std::__pop_heap (_RandomAccessIterator __first,  
_RandomAccessIterator __last, _RandomAccessIterator __result, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp> void std::__push_heap (_Random↵  
AccessIterator __first, _Distance __holeIndex, _Distance __topIndex, _Tp __value)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare> void std::__push_↵  
heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __topIndex, _Tp __value, _Compare  
__comp)`
- `template<typename _RandomAccessIterator> bool std::is_heap (_RandomAccessIterator __first, _RandomAccess↵  
Iterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare> bool std::is_heap (_RandomAccessIterator __first, ↵  
RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator> _RandomAccessIterator std::is_heap_until (_RandomAccessIterator __↵  
first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare> _RandomAccessIterator std::is_heap_until (↵  
RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator> void std::make_heap (_RandomAccessIterator __first, _RandomAccess↵  
Iterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare> void std::make_heap (_RandomAccessIterator __first,  
_RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator> void std::pop_heap (_RandomAccessIterator __first, _RandomAccess↵  
Iterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare> void std::pop_heap (_RandomAccessIterator __first,  
_RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator> void std::push_heap (_RandomAccessIterator __first, _RandomAccess↵  
Iterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare> void std::push_heap (_RandomAccessIterator __first,  
_RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator> void std::sort_heap (_RandomAccessIterator __first, _RandomAccess↵  
Iterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare> void std::sort_heap (_RandomAccessIterator __first,  
_RandomAccessIterator __last, _Compare __comp)`

### 5.393.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<queue>`.

## 5.394 `std_iterator.h` File Reference

### Classes

- class `std::back_insert_iterator< _Container >`
- class `std::front_insert_iterator< _Container >`
- class `std::insert_iterator< _Container >`
- class `std::move_iterator< _Iterator >`
- class `std::reverse_iterator< _Iterator >`



## Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

## Macros

- `#define _GLIBCXX_MAKE_MOVE_IF_NOEXCEPT_ITERATOR(_Iter)`
- `#define _GLIBCXX_MAKE_MOVE_ITERATOR(_Iter)`

## Functions

- `template<typename _Iterator, typename _ReturnType = typename conditional<__move_if_noexcept_cond <typename iterator_traits<_Iterator>::value_type>::value, _Iterator, move_iterator<_Iterator>>::type> _ReturnType std::__make_move_if_noexcept_iterator(_Iterator __i)`
- `template<typename _Container> back_insert_iterator<_Container> std::back_inserter(_Container &__x)`
- `template<typename _Container> front_insert_iterator<_Container> std::front_inserter(_Container &__x)`
- `template<typename _Container, typename _Iterator> insert_iterator<_Container> std::inserter(_Container &__x, _Iterator __i)`
- `template<typename _Iterator> move_iterator<_Iterator> std::make_move_iterator(_Iterator __i)`
- `template<typename _Iterator> bool std::operator!= (const reverse_iterator<_Iterator> &__x, const reverse_iterator<_Iterator> &__y)`
- `template<typename _IteratorL, typename _IteratorR> bool std::operator!= (const reverse_iterator<_IteratorL> &__x, const reverse_iterator<_IteratorR> &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container> bool __gnu_cxx::operator!= (const __normal_iterator<_IteratorL, _Container> &__lhs, const __normal_iterator<_IteratorR, _Container> &__rhs)`
- `template<typename _Iterator, typename _Container> bool __gnu_cxx::operator!= (const __normal_iterator<_Iterator, _Container> &__lhs, const __normal_iterator<_Iterator, _Container> &__rhs)`
- `template<typename _IteratorL, typename _IteratorR> bool std::operator!= (const move_iterator<_IteratorL> &__x, const move_iterator<_IteratorR> &__y)`
- `template<typename _Iterator> bool std::operator!= (const move_iterator<_Iterator> &__x, const move_iterator<_Iterator> &__y)`
- `template<typename _Iterator> reverse_iterator<_Iterator> std::operator+ (typename reverse_iterator<_Iterator>::difference_type __n, const reverse_iterator<_Iterator> &__x)`
- `template<typename _Iterator, typename _Container> __normal_iterator<_Iterator, _Container> __gnu_cxx::operator+ (typename __normal_iterator<_Iterator, _Container>::difference_type __n, const __normal_iterator<_Iterator, _Container> &__i)`
- `template<typename _Iterator> move_iterator<_Iterator> std::operator+ (typename move_iterator<_Iterator>::difference_type __n, const move_iterator<_Iterator> &__x)`
- `template<typename _Iterator> reverse_iterator<_Iterator>::difference_type std::operator- (const reverse_iterator<_Iterator> &__x, const reverse_iterator<_Iterator> &__y)`
- `template<typename _IteratorL, typename _IteratorR> auto std::operator- (const reverse_iterator<_IteratorL> &__x, const reverse_iterator<_IteratorR> &__y) -> decltype(__y.base()-__x.base())`
- `template<typename _IteratorL, typename _IteratorR, typename _Container> auto __gnu_cxx::operator- (const __normal_iterator<_IteratorL, _Container> &__lhs, const __normal_iterator<_IteratorR, _Container> &__rhs) -> decltype(__lhs.base()-__rhs.base())`
- `template<typename _Iterator, typename _Container> __normal_iterator<_Iterator, _Container>::difference_type __gnu_cxx::operator- (const __normal_iterator<_Iterator, _Container> &__lhs, const __normal_iterator<_Iterator, _Container> &__rhs)`
- `template<typename _IteratorL, typename _IteratorR> auto std::operator- (const move_iterator<_IteratorL> &__x, const move_iterator<_IteratorR> &__y) -> decltype(__x.base()-__y.base())`

- Generated on Tue Jun 30 2015 12:19:40 for libstdc++ by Doxygen

- `template<typename _IteratorL, typename _IteratorR> bool std::operator>= (const reverse_iterator< _IteratorL> &__x, const reverse_iterator< _IteratorR> &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container> bool __gnu_cxx::operator>= (const __↵ normal_iterator< _IteratorL, _Container> &__lhs, const __normal_iterator< _IteratorR, _Container> &__rhs)`
- `template<typename _Iterator, typename _Container> bool __gnu_cxx::operator>= (const __normal_iterator< _Iterator, _Container> &__lhs, const __normal_iterator< _Iterator, _Container> &__rhs)`
- `template<typename _IteratorL, typename _IteratorR> bool std::operator>= (const move_iterator< _IteratorL> &__x, const move_iterator< _IteratorR> &__y)`
- `template<typename _Iterator> bool std::operator>= (const move_iterator< _Iterator> &__x, const move_iterator< _Iterator> &__y)`

#### 5.394.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

This file implements `reverse_iterator`, `back_insert_iterator`, `front_insert_iterator`, `insert_iterator`, `__normal_iterator`, and their supporting functions and overloaded operators.

### 5.395 `stl_iterator_base_funcs.h` File Reference

#### Namespaces

- [std](#)

#### Functions

- `template<typename _InputIterator, typename _Distance> void std::__advance (_InputIterator &__i, _Distance __n, input↵ _iterator_tag)`
- `template<typename _BidirectionalIterator, typename _Distance> void std::__advance (_BidirectionalIterator &__i, ↵ Distance __n, bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Distance> void std::__advance (_RandomAccessIterator &__i, ↵ _Distance __n, random_access_iterator_tag)`
- `template<typename _InputIterator> iterator_traits< _InputIterator>::difference_type std::__distance (_InputIterator ↵ __first, _InputIterator __last, input_iterator_tag)`
- `template<typename _RandomAccessIterator> iterator_traits< _RandomAccessIterator>::difference_type std::__↵ distance (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _InputIterator, typename _Distance> void std::advance (_InputIterator &__i, _Distance __n)`
- `template<typename _InputIterator> iterator_traits< _InputIterator>::difference_type std::distance (_InputIterator ↵ __first, _InputIterator __last)`
- `template<typename _ForwardIterator> _ForwardIterator std::next (_ForwardIterator __x, typename iterator_traits< ↵ _ForwardIterator>::difference_type __n=1)`
- `template<typename _BidirectionalIterator> _BidirectionalIterator std::prev (_BidirectionalIterator __x, ↵ typename iterator_traits< _BidirectionalIterator>::difference_type __n=1)`

#### 5.395.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

This file contains all of the general iterator-related utility functions, such as `distance()` and `advance()`.

## 5.396 `std_iterator_base_types.h` File Reference

### Classes

- class `std::__has_iterator_category_helper< _Tp >`
- struct `std::bidirectional_iterator_tag`
- struct `std::forward_iterator_tag`
- struct `std::input_iterator_tag`
- struct `std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >`
- struct `std::iterator_traits< _Tp * >`
- struct `std::iterator_traits< const _Tp * >`
- struct `std::output_iterator_tag`
- struct `std::random_access_iterator_tag`

### Namespaces

- `std`

### Typedefs

- `template<typename _InIter > using std::RequireInputIter = typename enable_if< is_convertible< typename iterator_traits< _InIter >::iterator_category, input_iterator_tag >::value >::type`

### Functions

- `template<typename _Iter > iterator_traits< _Iter >::iterator_category std::__iterator_category (const _Iter &)`

#### 5.396.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

This file contains all of the general iterator-related utility types, such as `iterator_traits` and `struct iterator`.

## 5.397 `std_list.h` File Reference

### Classes

- struct `std::__detail::_List_node_base`
- class `std::_List_base< _Tp, _Alloc >`
- struct `std::_List_const_iterator< _Tp >`
- struct `std::_List_iterator< _Tp >`
- struct `std::_List_node< _Tp >`
- class `std::list< _Tp, _Alloc >`

### Namespaces

- `std`
- `std::__detail`

## Functions

- `template<typename _Val > bool std::operator!= (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y)`
- `template<typename _Tp, typename _Alloc > bool std::operator!= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc > bool std::operator< (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc > bool std::operator<= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Val > bool std::operator== (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y)`
- `template<typename _Tp, typename _Alloc > bool std::operator== (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc > bool std::operator> (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc > bool std::operator>= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc > void std::swap (list< _Tp, _Alloc > &__x, list< _Tp, _Alloc > &__y)`

### 5.397.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<list>`.

## 5.398 `std::map.h` File Reference

### Classes

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

### Namespaces

- `std`

### Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc > bool std::operator!= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc > bool std::operator< (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc > bool std::operator<= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc > bool std::operator== (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc > bool std::operator> (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc > bool std::operator>= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc > void std::swap (map< _Key, _Tp, _Compare, _Alloc > &__x, map< _Key, _Tp, _Compare, _Alloc > &__y)`

## 5.398.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map>`.

5.399 `std::multimap.h` File Reference

## Classes

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

## Namespaces

- `std`

## Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool std::operator!= (const multimap<_Key, _Tp, _Compare, _Alloc> &__x, const multimap<_Key, _Tp, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool std::operator< (const multimap<_Key, _Tp, _Compare, _Alloc> &__x, const multimap<_Key, _Tp, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool std::operator<= (const multimap<_Key, _Tp, _Compare, _Alloc> &__x, const multimap<_Key, _Tp, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool std::operator== (const multimap<_Key, _Tp, _Compare, _Alloc> &__x, const multimap<_Key, _Tp, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool std::operator> (const multimap<_Key, _Tp, _Compare, _Alloc> &__x, const multimap<_Key, _Tp, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool std::operator>= (const multimap<_Key, _Tp, _Compare, _Alloc> &__x, const multimap<_Key, _Tp, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> void std::swap (multimap<_Key, _Tp, _Compare, _Alloc> &__x, multimap<_Key, _Tp, _Compare, _Alloc> &__y)`

## 5.399.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map>`.

5.400 `std::multiset.h` File Reference

## Classes

- class `std::multiset<_Key, _Compare, _Alloc>`

## Namespaces

- `std`

## Functions

- `template<typename _Key, typename _Compare, typename _Alloc> bool std::operator!= (const multiset< _Key, _Compare, _Alloc> &__x, const multiset< _Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc> bool std::operator< (const multiset< _Key, _Compare, _Alloc> &__x, const multiset< _Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc> bool std::operator<= (const multiset< _Key, _Compare, _Alloc> &__x, const multiset< _Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc> bool std::operator== (const multiset< _Key, _Compare, _Alloc> &__x, const multiset< _Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc> bool std::operator> (const multiset< _Key, _Compare, _Alloc> &__x, const multiset< _Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc> bool std::operator>= (const multiset< _Key, _Compare, _Alloc> &__x, const multiset< _Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc> void std::swap (multiset< _Key, _Compare, _Alloc> &__x, multiset< _Key, _Compare, _Alloc> &__y)`

### 5.400.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<set>`.

## 5.401 `stl_numeric.h` File Reference

### Namespaces

- [std](#)

### Functions

- `template<typename _InputIterator, typename _Tp> _Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation> _Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator> _OutputIterator std::adjacent\_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation> _OutputIterator std::adjacent\_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp> _Tp std::inner\_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2> _Tp std::inner\_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)`
- `template<typename _ForwardIterator, typename _Tp> void std::iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)`
- `template<typename _InputIterator, typename _OutputIterator> _OutputIterator std::partial\_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation> _OutputIterator std::partial\_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`

## 5.401.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<numeric>`.

5.402 `std_pair.h` File Reference

## Classes

- struct [std::pair<\\_T1, \\_T2>](#)
- struct [std::piecewise\\_construct\\_t](#)

## Namespaces

- [std](#)

## Functions

- `template<class _T1, class _T2> constexpr pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_strip< _T2 >::__type> std::make\_pair (_T1 &&__x, _T2 &&__y)`
- `template<class _T1, class _T2> constexpr bool std::operator!= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2> constexpr bool std::operator< (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2> constexpr bool std::operator<= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2> constexpr bool std::operator== (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2> constexpr bool std::operator> (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2> constexpr bool std::operator>= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2> void std::swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y) noexcept(noexcept(__x.swap(__y)))`

## Variables

- `constexpr piecewise_construct_t std::piecewise\_construct`

## 5.402.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

5.403 `std_queue.h` File Reference

## Classes

- class [std::priority\\_queue<\\_Tp, \\_Sequence, \\_Compare>](#)
- class [std::queue<\\_Tp, \\_Sequence>](#)



## Namespaces

- [std](#)

## Functions

- `template<typename _Tp, typename _Seq> bool std::operator!= (const queue< _Tp, _Seq> &__x, const queue< _Tp, _Seq> &__y)`
- `template<typename _Tp, typename _Seq> bool std::operator< (const queue< _Tp, _Seq> &__x, const queue< _Tp, _Seq> &__y)`
- `template<typename _Tp, typename _Seq> bool std::operator<= (const queue< _Tp, _Seq> &__x, const queue< _Tp, _Seq> &__y)`
- `template<typename _Tp, typename _Seq> bool std::operator== (const queue< _Tp, _Seq> &__x, const queue< _Tp, _Seq> &__y)`
- `template<typename _Tp, typename _Seq> bool std::operator> (const queue< _Tp, _Seq> &__x, const queue< _Tp, _Seq> &__y)`
- `template<typename _Tp, typename _Seq> bool std::operator>= (const queue< _Tp, _Seq> &__x, const queue< _Tp, _Seq> &__y)`
- `template<typename _Tp, typename _Seq> void std::swap (queue< _Tp, _Seq> &__x, queue< _Tp, _Seq> &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, typename _Sequence, typename _Compare> void std::swap (priority_queue< _Tp, _Sequence, _Compare> &__x, priority_queue< _Tp, _Sequence, _Compare> &__y) noexcept(noexcept(__x.swap(__y)))`

## 5.403.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<queue>`.

5.404 `stl_raw_storage_iter.h` File Reference

## Classes

- class [std::raw\\_storage\\_iterator< \\_OutputIterator, \\_Tp>](#)

## Namespaces

- [std](#)

## 5.404.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

5.405 `stl_relops.h` File Reference

## Namespaces

- [std](#)
- [std::rel\\_ops](#)

## Functions

- `template<class _Tp> bool std::rel_ops::operator!= (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp> bool std::rel_ops::operator<= (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp> bool std::rel_ops::operator> (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp> bool std::rel_ops::operator>= (const _Tp &__x, const _Tp &__y)`

### 5.405.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

Inclusion of this file has been removed from all of the other STL headers for safety reasons, except `std_` utility.h. For more information, see the thread of about twenty messages starting with <http://gcc.gnu.org/ml/libstdc++/2001-01/msg00223.html>, or [http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#faq.ambiguous\\_overloads](http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#faq.ambiguous_overloads)

Short summary: the `rel_ops` operators should be avoided for the present.

## 5.406 `std_set.h` File Reference

### Classes

- class `std::set<_Key, _Compare, _Alloc>`

### Namespaces

- `std`

## Functions

- `template<typename _Key, typename _Compare, typename _Alloc> bool std::operator!= (const set<_Key, _Compare, _Alloc> &__x, const set<_Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc> bool std::operator< (const set<_Key, _Compare, _Alloc> &__x, const set<_Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc> bool std::operator<= (const set<_Key, _Compare, _Alloc> &__x, const set<_Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc> bool std::operator== (const set<_Key, _Compare, _Alloc> &__x, const set<_Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc> bool std::operator> (const set<_Key, _Compare, _Alloc> &__x, const set<_Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc> bool std::operator>= (const set<_Key, _Compare, _Alloc> &__x, const set<_Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc> void std::swap (set<_Key, _Compare, _Alloc> &__x, set<_Key, _Compare, _Alloc> &__y)`

### 5.406.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<set>`.

## 5.407 `std::stack.h` File Reference

### Classes

- class [std::stack<\\_Tp, \\_Sequence>](#)

### Namespaces

- [std](#)

### Functions

- template<typename \_Tp, typename \_Seq> bool [std::operator!=](#) (const stack<\_Tp, \_Seq> &\_\_x, const stack<\_Tp, \_Seq> &\_\_y)
- template<typename \_Tp, typename \_Seq> bool [std::operator<](#) (const stack<\_Tp, \_Seq> &\_\_x, const stack<\_Tp, \_Seq> &\_\_y)
- template<typename \_Tp, typename \_Seq> bool [std::operator<=](#) (const stack<\_Tp, \_Seq> &\_\_x, const stack<\_Tp, \_Seq> &\_\_y)
- template<typename \_Tp, typename \_Seq> bool [std::operator==](#) (const stack<\_Tp, \_Seq> &\_\_x, const stack<\_Tp, \_Seq> &\_\_y)
- template<typename \_Tp, typename \_Seq> bool [std::operator>](#) (const stack<\_Tp, \_Seq> &\_\_x, const stack<\_Tp, \_Seq> &\_\_y)
- template<typename \_Tp, typename \_Seq> bool [std::operator>=](#) (const stack<\_Tp, \_Seq> &\_\_x, const stack<\_Tp, \_Seq> &\_\_y)
- template<typename \_Tp, typename \_Seq> void [std::swap](#) (stack<\_Tp, \_Seq> &\_\_x, stack<\_Tp, \_Seq> &\_\_y) noexcept(noexcept(\_\_x.swap(\_\_y)))

### 5.407.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<stack>`.

## 5.408 `std::tempbuf.h` File Reference

### Classes

- class [std::\\_Temporary\\_buffer<\\_ForwardIterator, \\_Tp>](#)

### Namespaces

- [std](#)

### Functions

- template<typename \_Pointer, typename \_ForwardIterator> void [std::\\_\\_uninitialized\\_construct\\_buf](#) (\_Pointer \_\_first, \_Pointer \_\_last, \_ForwardIterator \_\_seed)
- template<typename \_Tp> pair<\_Tp\*, ptrdiff\_t> [std::get\\_temporary\\_buffer](#) (ptrdiff\_t \_\_len) noexcept
- template<typename \_Tp> void [std::return\\_temporary\\_buffer](#) (\_Tp \*\_\_p)

## 5.408.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

5.409 `std_tree.h` File Reference

## Namespaces

- [std](#)

## Enumerations

- enum `_Rb_tree_color` { `_S_red`, `_S_black` }

## Functions

- unsigned int **`std::Rb_tree_black_count`** (const `_Rb_tree_node_base *``__node`, const `_Rb_tree_node_base *``__root`) throw ()
- `_Rb_tree_node_base *` **`std::Rb_tree_decrement`** (`_Rb_tree_node_base *``__x`) throw ()
- const `_Rb_tree_node_base *` **`std::Rb_tree_decrement`** (const `_Rb_tree_node_base *``__x`) throw ()
- `_Rb_tree_node_base *` **`std::Rb_tree_increment`** (`_Rb_tree_node_base *``__x`) throw ()
- const `_Rb_tree_node_base *` **`std::Rb_tree_increment`** (const `_Rb_tree_node_base *``__x`) throw ()
- void **`std::Rb_tree_insert_and_rebalance`** (const bool `__insert_left`, `_Rb_tree_node_base *``__x`, `_Rb_tree_node_base *``__p`, `_Rb_tree_node_base &``__header`) throw ()
- `_Rb_tree_node_base *` **`std::Rb_tree_rebalance_for_erase`** (`_Rb_tree_node_base *``__z`, `_Rb_tree_node_base &``__header`) throw ()
- template<typename `_Val` > bool **`std::operator!=`** (const `_Rb_tree_iterator`< `_Val` > &`__x`, const `_Rb_tree_const_iterator`< `_Val` > &`__y`)
- template<typename `_Key`, typename `_Val`, typename `_KeyOfValue`, typename `_Compare`, typename `_Alloc` > bool **`std::operator!=`** (const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &`__x`, const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &`__y`)
- template<typename `_Key`, typename `_Val`, typename `_KeyOfValue`, typename `_Compare`, typename `_Alloc` > bool **`std::operator<`** (const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &`__x`, const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &`__y`)
- template<typename `_Key`, typename `_Val`, typename `_KeyOfValue`, typename `_Compare`, typename `_Alloc` > bool **`std::operator<=`** (const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &`__x`, const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &`__y`)
- template<typename `_Val` > bool **`std::operator==`** (const `_Rb_tree_iterator`< `_Val` > &`__x`, const `_Rb_tree_const_iterator`< `_Val` > &`__y`)
- template<typename `_Key`, typename `_Val`, typename `_KeyOfValue`, typename `_Compare`, typename `_Alloc` > bool **`std::operator==`** (const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &`__x`, const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &`__y`)
- template<typename `_Key`, typename `_Val`, typename `_KeyOfValue`, typename `_Compare`, typename `_Alloc` > bool **`std::operator>`** (const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &`__x`, const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &`__y`)
- template<typename `_Key`, typename `_Val`, typename `_KeyOfValue`, typename `_Compare`, typename `_Alloc` > bool **`std::operator>=`** (const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &`__x`, const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &`__y`)
- template<typename `_Key`, typename `_Val`, typename `_KeyOfValue`, typename `_Compare`, typename `_Alloc` > void **`std::swap`** (`_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &`__x`, `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &`__y`)

### 5.409.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map>` or `<set>`.

## 5.410 `std_uninitialized.h` File Reference

### Namespaces

- [std](#)

### Functions

- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator> _ForwardIterator std::__uninitialized_copy_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp> _ForwardIterator std::__uninitialized_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, allocator<_Tp> &__a)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, typename _Allocator> _ForwardIterator std::__uninitialized_copy_move (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator> _ForwardIterator std::__uninitialized_copy_n (_InputIterator __first, _Size __n, _ForwardIterator __result, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Size, typename _ForwardIterator> _ForwardIterator std::__uninitialized_copy_n (_RandomAccessIterator __first, _Size __n, _ForwardIterator __result, random_access_iterator_tag)`
- `template<typename _ForwardIterator> void std::__uninitialized_default (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Allocator> void std::__uninitialized_default_a (_ForwardIterator __first, _ForwardIterator __last, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Tp> void std::__uninitialized_default_a (_ForwardIterator __first, _ForwardIterator __last, allocator<_Tp> &__a)`
- `template<typename _ForwardIterator, typename _Size> void std::__uninitialized_default_n (_ForwardIterator __first, _Size __n)`
- `template<typename _ForwardIterator, typename _Size, typename _Allocator> void std::__uninitialized_default_n_a (_ForwardIterator __first, _Size __n, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp> void std::__uninitialized_default_n_a (_ForwardIterator __first, _Size __n, allocator<_Tp> &__a)`
- `template<typename _ForwardIterator, typename _Tp, typename _Allocator> void std::__uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Tp, typename _Tp2> void std::__uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x, allocator<_Tp2> &__a)`
- `template<typename _ForwardIterator, typename _Tp, typename _InputIterator, typename _Allocator> _ForwardIterator std::__uninitialized_fill_move (_ForwardIterator __result, _ForwardIterator __mid, const _Tp &__x, _InputIterator __first, _InputIterator __last, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Allocator> void std::__uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const _Tp &__x, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Tp2> void std::__uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const _Tp &__x, allocator<_Tp2> &__a)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator> _ForwardIterator std::__uninitialized_move_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)`

- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, typename _Allocator> _ForwardIterator std::uninitialized_move_copy (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp, typename _Allocator> void std::uninitialized_move_fill (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, const _Tp &__x, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator> _ForwardIterator std::uninitialized_move_if_noexcept_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator> _ForwardIterator std::uninitialized_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator> _ForwardIterator std::uninitialized_copy_n (_InputIterator __first, _Size __n, _ForwardIterator __result)`
- `template<typename _ForwardIterator, typename _Tp> void std::uninitialized_fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp> void std::uninitialized_fill_n (_ForwardIterator __first, _Size __n, const _Tp &__x)`

#### 5.410.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 5.411 `std_vector.h` File Reference

### Classes

- struct `std::_Vector_base< _Tp, _Alloc >`
- class `std::vector< _Tp, _Alloc >`

### Namespaces

- `std`

### Functions

- `template<typename _Tp, typename _Alloc> bool std::operator!= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc> bool std::operator< (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc> bool std::operator<= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc> bool std::operator== (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc> bool std::operator> (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc> bool std::operator>= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc> void std::swap (vector< _Tp, _Alloc > &__x, vector< _Tp, _Alloc > &__y)`

### 5.411.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<vector>`.

## 5.412 `stream_iterator.h` File Reference

### Classes

- class [std::istream\\_iterator<\\_Tp, \\_CharT, \\_Traits, \\_Dist>](#)
- class [std::ostream\\_iterator<\\_Tp, \\_CharT, \\_Traits>](#)

### Namespaces

- [std](#)

### Functions

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

### 5.412.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

## 5.413 `streambuf_iterator.h` File Reference

### Classes

- class [std::istreambuf\\_iterator<\\_CharT, \\_Traits>](#)
- class [std::ostreambuf\\_iterator<\\_CharT, \\_Traits>](#)

### Namespaces

- [std](#)

### Functions

- `template<bool _IsMove, typename _CharT> __gnu_cxx::__enable_if<__is_char<_CharT>::__value, ostreambuf_iterator<_CharT>>::__type std::\_\_copy\_move\_a2 (_CharT *__first, _CharT *__last, ostreambuf_iterator<_CharT> __result)`
- `template<bool _IsMove, typename _CharT> __gnu_cxx::__enable_if<__is_char<_CharT>::__value, ostreambuf_iterator<_CharT>>::__type std::\_\_copy\_move\_a2 (const _CharT *__first, const _CharT *__last, ostreambuf_iterator<_CharT> __result)`

- `template<bool _IsMove, typename _CharT > __gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type std::copy_move_a2 (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT * __result)`
- `template<typename _CharT > __gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::copy (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT > __result)`
- `template<typename _CharT > __gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf_iterator< _CharT > >::__type std::find (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT & __val)`
- `template<typename _CharT, typename _Traits > bool std::operator!= (const istreambuf_iterator< _CharT, _Traits > & __a, const istreambuf_iterator< _CharT, _Traits > & __b)`
- `template<typename _CharT, typename _Traits > bool std::operator== (const istreambuf_iterator< _CharT, _Traits > & __a, const istreambuf_iterator< _CharT, _Traits > & __b)`

#### 5.413.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

## 5.414 string\_conversions.h File Reference

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### Functions

- `template<typename _TRet, typename _Ret = _TRet, typename _CharT, typename... _Base> _Ret __gnu_cxx::__stoa (_TRet(*__convf)(const _CharT *, _CharT **, _Base...), const char * __name, const _CharT * __str, std::size_t * __idx, _Base... __base)`
- `template<typename _String, typename _CharT = typename _String::value_type> _String __gnu_cxx::__to_xstring (int(*__convf)(_CharT *, std::size_t, const _CharT *, __builtin_va_list), std::size_t __n, const _CharT * __fmt,...)`

#### 5.414.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.415 stringfwd.h File Reference

### Classes

- class [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >](#)
- struct [std::char\\_traits< \\_CharT >](#)

### Namespaces

- [std](#)



## Typedefs

- typedef basic\_string< char > [std::string](#)
- typedef basic\_string< char16\_t > [std::u16string](#)
- typedef basic\_string< char32\_t > [std::u32string](#)
- typedef basic\_string< wchar\_t > [std::wstring](#)

### 5.415.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

## 5.416 synth\_access\_traits.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::synth\\_access\\_traits< Type\\_Traits, Set, \\_ATraits >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- #define **PB\_DS\_SYNTH\_E\_ACCESS\_TRAITS\_C\_DEC**
- #define **PB\_DS\_SYNTH\_E\_ACCESS\_TRAITS\_T\_DEC**

### 5.416.1 Detailed Description

Contains an implementation class for a patricia tree.

## 5.417 tag\_and\_trait.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::associative\\_tag](#)
- struct [\\_\\_gnu\\_pbds::basic\\_branch\\_tag](#)
- struct [\\_\\_gnu\\_pbds::basic\\_hash\\_tag](#)
- struct [\\_\\_gnu\\_pbds::basic\\_invalidation\\_guarantee](#)
- struct [\\_\\_gnu\\_pbds::binary\\_heap\\_tag](#)
- struct [\\_\\_gnu\\_pbds::binomial\\_heap\\_tag](#)
- struct [\\_\\_gnu\\_pbds::cc\\_hash\\_tag](#)
- struct [\\_\\_gnu\\_pbds::container\\_tag](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits< Cntnr >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< \\_Tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< binary\\_heap\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< binomial\\_heap\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< cc\\_hash\\_tag >](#)

- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< gp\\_hash\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< list\\_update\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< ov\\_tree\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< pairing\\_heap\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< pat\\_trie\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< rb\\_tree\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< rc\\_binomial\\_heap\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< splay\\_tree\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< thin\\_heap\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, Tag, Policy\\_Tl >](#)
- struct [\\_\\_gnu\\_pbds::gp\\_hash\\_tag](#)
- struct [\\_\\_gnu\\_pbds::list\\_update\\_tag](#)
- struct [\\_\\_gnu\\_pbds::null\\_node\\_update< \\_Tp1, \\_Tp2, \\_Tp3, \\_Tp4 >](#)
- struct [\\_\\_gnu\\_pbds::null\\_type](#)
- struct [\\_\\_gnu\\_pbds::ov\\_tree\\_tag](#)
- struct [\\_\\_gnu\\_pbds::pairing\\_heap\\_tag](#)
- struct [\\_\\_gnu\\_pbds::pat\\_trie\\_tag](#)
- struct [\\_\\_gnu\\_pbds::point\\_invalidation\\_guarantee](#)
- struct [\\_\\_gnu\\_pbds::priority\\_queue\\_tag](#)
- struct [\\_\\_gnu\\_pbds::range\\_invalidation\\_guarantee](#)
- struct [\\_\\_gnu\\_pbds::rb\\_tree\\_tag](#)
- struct [\\_\\_gnu\\_pbds::rc\\_binomial\\_heap\\_tag](#)
- struct [\\_\\_gnu\\_pbds::sequence\\_tag](#)
- struct [\\_\\_gnu\\_pbds::splay\\_tree\\_tag](#)
- struct [\\_\\_gnu\\_pbds::string\\_tag](#)
- struct [\\_\\_gnu\\_pbds::thin\\_heap\\_tag](#)
- struct [\\_\\_gnu\\_pbds::tree\\_tag](#)
- struct [\\_\\_gnu\\_pbds::trie\\_tag](#)
- struct [\\_\\_gnu\\_pbds::trivial\\_iterator\\_tag](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Typedefs

- typedef void [\\_\\_gnu\\_pbds::trivial\\_iterator\\_difference\\_type](#)

### 5.417.1 Detailed Description

Contains tags and traits, e.g., ones describing underlying data structures.

## 5.418 tags.h File Reference

### Classes

- struct [\\_\\_gnu\\_parallel::balanced\\_quicksort\\_tag](#)
- struct [\\_\\_gnu\\_parallel::balanced\\_tag](#)
- struct [\\_\\_gnu\\_parallel::constant\\_size\\_blocks\\_tag](#)

- [struct \\_\\_gnu\\_parallel::default\\_parallel\\_tag](#)
- [struct \\_\\_gnu\\_parallel::equal\\_split\\_tag](#)
- [struct \\_\\_gnu\\_parallel::exact\\_tag](#)
- [struct \\_\\_gnu\\_parallel::find\\_tag](#)
- [struct \\_\\_gnu\\_parallel::growing\\_blocks\\_tag](#)
- [struct \\_\\_gnu\\_parallel::multiway\\_mergesort\\_exact\\_tag](#)
- [struct \\_\\_gnu\\_parallel::multiway\\_mergesort\\_sampling\\_tag](#)
- [struct \\_\\_gnu\\_parallel::multiway\\_mergesort\\_tag](#)
- [struct \\_\\_gnu\\_parallel::omp\\_loop\\_static\\_tag](#)
- [struct \\_\\_gnu\\_parallel::omp\\_loop\\_tag](#)
- [struct \\_\\_gnu\\_parallel::parallel\\_tag](#)
- [struct \\_\\_gnu\\_parallel::quicksort\\_tag](#)
- [struct \\_\\_gnu\\_parallel::sampling\\_tag](#)
- [struct \\_\\_gnu\\_parallel::sequential\\_tag](#)
- [struct \\_\\_gnu\\_parallel::unbalanced\\_tag](#)

#### Namespaces

- [\\_\\_gnu\\_parallel](#)

#### 5.418.1 Detailed Description

Tags for compile-time selection. This file is a GNU parallel extension to the Standard C++ Library.

### 5.419 `tgmath.h` File Reference

#### Macros

- `#define _GLIBCXX_TGMATH_H`

#### 5.419.1 Detailed Description

This is a Standard C++ Library header.

### 5.420 `thin_heap.hpp` File Reference

#### Classes

- [class \\_\\_gnu\\_pbds::detail::thin\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_ASSERT_AUX_NULL(X)`
- `#define PB_DS_ASSERT_NODE_CONSISTENT(_Node, _Bool)`
- `#define PB_DS_BASE_T_P`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

## Enumerations

- enum { `num_distinct_rank_bounds` }

## Variables

- static const `std::size_t __gnu_pbds::detail::g_a_rank_bounds` [`num_distinct_rank_bounds`]

## 5.420.1 Detailed Description

Contains an implementation class for a thin heap.

5.421 `throw_allocator.h` File Reference

## Classes

- struct [\\_\\_gnu\\_cxx::annotate\\_base](#)
- struct [\\_\\_gnu\\_cxx::condition\\_base](#)
- struct [\\_\\_gnu\\_cxx::forced\\_error](#)
- struct [\\_\\_gnu\\_cxx::limit\\_condition](#)
- struct [\\_\\_gnu\\_cxx::limit\\_condition::always\\_adjustor](#)
- struct [\\_\\_gnu\\_cxx::limit\\_condition::limit\\_adjustor](#)
- struct [\\_\\_gnu\\_cxx::limit\\_condition::never\\_adjustor](#)
- struct [\\_\\_gnu\\_cxx::random\\_condition](#)
- struct [\\_\\_gnu\\_cxx::random\\_condition::always\\_adjustor](#)
- struct [\\_\\_gnu\\_cxx::random\\_condition::group\\_adjustor](#)
- struct [\\_\\_gnu\\_cxx::random\\_condition::never\\_adjustor](#)
- class [\\_\\_gnu\\_cxx::throw\\_allocator\\_base< \\_Tp, \\_Cond >](#)
- struct [\\_\\_gnu\\_cxx::throw\\_allocator\\_limit< \\_Tp >](#)
- struct [\\_\\_gnu\\_cxx::throw\\_allocator\\_random< \\_Tp >](#)
- struct [\\_\\_gnu\\_cxx::throw\\_value\\_base< \\_Cond >](#)
- struct [\\_\\_gnu\\_cxx::throw\\_value\\_limit](#)
- struct [\\_\\_gnu\\_cxx::throw\\_value\\_random](#)
- struct [std::hash< \\_\\_gnu\\_cxx::throw\\_value\\_limit >](#)
- struct [std::hash< \\_\\_gnu\\_cxx::throw\\_value\\_random >](#)

## Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

## Functions

- `void __gnu_cxx::__throw_forced_error ()`
- `template<typename _Tp, typename _Cond > bool __gnu_cxx::operator!= (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)`
- `template<typename _Cond > throw_value_base< _Cond > __gnu_cxx::operator* (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _Cond > throw_value_base< _Cond > __gnu_cxx::operator+ (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _Cond > throw_value_base< _Cond > __gnu_cxx::operator- (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _Cond > bool __gnu_cxx::operator< (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `std::ostream & __gnu_cxx::operator<< (std::ostream &os, const annotate_base &__b)`
- `template<typename _Cond > bool __gnu_cxx::operator== (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _Tp, typename _Cond > bool __gnu_cxx::operator== (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)`
- `template<typename _Cond > void __gnu_cxx::swap (throw_value_base< _Cond > &__a, throw_value_base< _Cond > &__b)`

### 5.421.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Contains two exception-generating types (`throw_value`, `throw_allocator`) intended to be used as value and allocator types while testing exception safety in templated containers and algorithms. The allocator has additional log and debug features. The exception generated is of type `forced_exception_error`.

## 5.422 `time_members.h` File Reference

### Namespaces

- [std](#)

### 5.422.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 5.423 `trace_fn_imps.hpp` File Reference

### 5.423.1 Detailed Description

Contains an implementation class for a `binary_heap`.

## 5.424 `trace_fn_imps.hpp` File Reference

### 5.424.1 Detailed Description

Contains implementations of `cc_ht_map_`'s trace-mode functions.

## 5.425 `trace_fn_imps.hpp` File Reference

### 5.425.1 Detailed Description

Contains implementations of `gp_ht_map_`'s trace-mode functions.

## 5.426 `trace_fn_imps.hpp` File Reference

### 5.426.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

## 5.427 `trace_fn_imps.hpp` File Reference

### 5.427.1 Detailed Description

Contains implementations of `lu_map_`.

## 5.428 `trace_fn_imps.hpp` File Reference

### 5.428.1 Detailed Description

Contains an implementation class for `pat_trie_`.

## 5.429 `trace_fn_imps.hpp` File Reference

### 5.429.1 Detailed Description

Contains an implementation for `rc_binomial_heap_`.

## 5.430 `trace_fn_imps.hpp` File Reference

### 5.430.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

## 5.431 `traits.hpp` File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, Node, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_traits< Key, null\\_type, Cmp\\_Fn, Node\\_Update, Node, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.431.1 Detailed Description

Contains an implementation for `bin_search_tree_`.

### 5.432 `traits.hpp` File Reference

#### Classes

- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, Data, Cmp\\_Fn, Node\\_Update, Tag, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::trie\\_traits< Key, Data, \\_ATraits, Node\\_Update, Tag, \\_Alloc >](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### Macros

- `#define PB_DS_DEBUG_VERIFY(_Cond)`

#### 5.432.1 Detailed Description

Contains an implementation class for tree-like classes.

### 5.433 `traits.hpp` File Reference

#### Classes

- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, ov\\_tree\\_tag, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, null\\_type, Cmp\\_Fn, Node\\_Update, ov\\_tree\\_tag, \\_Alloc >](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.433.1 Detailed Description

Contains an implementation class for `ov_tree_`.

### 5.434 `traits.hpp` File Reference

#### Classes

- struct [\\_\\_gnu\\_pbds::detail::trie\\_traits< Key, Mapped, \\_ATraits, Node\\_Update, pat\\_trie\\_tag, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::trie\\_traits< Key, null\\_type, \\_ATraits, Node\\_Update, pat\\_trie\\_tag, \\_Alloc >](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

## 5.434.1 Detailed Description

Contains an implementation class for pat\_trie\_.

## 5.435 traits.hpp File Reference

## Classes

- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, rb\\_tree\\_tag, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, null\\_type, Cmp\\_Fn, Node\\_Update, rb\\_tree\\_tag, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## 5.435.1 Detailed Description

Contains an implementation for rb\_tree\_.

## 5.436 traits.hpp File Reference

## Classes

- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, splay\\_tree\\_tag, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, null\\_type, Cmp\\_Fn, Node\\_Update, splay\\_tree\\_tag, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## 5.436.1 Detailed Description

Contains an implementation for splay\_tree\_.

## 5.437 tree\_policy.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::tree\\_order\\_statistics\\_node\\_update< Node\\_Cltr, Node\\_Itr, Cmp\\_Fn, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)



## Macros

- `#define PB_DS_BRANCH_POLICY_BASE`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

### 5.437.1 Detailed Description

Contains tree-related policies.

## 5.438 tree\_trace\_base.hpp File Reference

### 5.438.1 Detailed Description

Contains tree-related policies.

## 5.439 trie\_policy.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::trie\\_order\\_statistics\\_node\\_update](#)< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >
- class [\\_\\_gnu\\_pbds::trie\\_prefix\\_search\\_node\\_update](#)< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >
- struct [\\_\\_gnu\\_pbds::trie\\_string\\_access\\_traits](#)< String, Min\_E\_Val, Max\_E\_Val, Reverse, \_Alloc >

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_TRIE_POLICY_BASE`

### 5.439.1 Detailed Description

Contains trie-related policies.

## 5.440 trie\_policy\_base.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::trie\\_policy\\_base](#)< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

## 5.440.1 Detailed Description

Contains an implementation of `trie_policy_base`.

5.441 `trie_string_access_traits_imp.hpp` File Reference

## 5.441.1 Detailed Description

Contains a policy for extracting character positions from a string for a vector-based PATRICIA tree

5.442 `type_traits.h` File Reference

## Namespaces

- [\\_\\_gnu\\_cxx](#)

## Functions

- `template<typename _Type> bool __gnu_cxx::__is_null_pointer (_Type * __ptr)`
- `template<typename _Type> bool __gnu_cxx::__is_null_pointer (_Type)`

## 5.442.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

5.443 `type_utils.hpp` File Reference

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_STATIC_ASSERT(UNIQUE, E)`

## Typedefs

- `typedef std::tr1::integral_constant< int, 0 > __gnu_pbds::detail::false_type`
- `typedef std::tr1::integral_constant< int, 1 > __gnu_pbds::detail::true_type`

### 5.443.1 Detailed Description

Contains utilities for handling types. All of these classes are based on Modern C++ by Andrei Alexandrescu.

## 5.444 typelist.h File Reference

### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [\\_\\_gnu\\_cxx::typelist](#)

### Macros

- `#define _GLIBCXX_TYPELIST_CHAIN1(X0)`
- `#define _GLIBCXX_TYPELIST_CHAIN10(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9)`
- `#define _GLIBCXX_TYPELIST_CHAIN11(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10)`
- `#define _GLIBCXX_TYPELIST_CHAIN12(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11)`
- `#define _GLIBCXX_TYPELIST_CHAIN13(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12)`
- `#define _GLIBCXX_TYPELIST_CHAIN14(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13)`
- `#define _GLIBCXX_TYPELIST_CHAIN15(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14)`
- `#define _GLIBCXX_TYPELIST_CHAIN16(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15)`
- `#define _GLIBCXX_TYPELIST_CHAIN17(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16)`
- `#define _GLIBCXX_TYPELIST_CHAIN18(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17)`
- `#define _GLIBCXX_TYPELIST_CHAIN19(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17, X18)`
- `#define _GLIBCXX_TYPELIST_CHAIN2(X0, X1)`
- `#define _GLIBCXX_TYPELIST_CHAIN20(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17, X18, X19)`
- `#define _GLIBCXX_TYPELIST_CHAIN3(X0, X1, X2)`
- `#define _GLIBCXX_TYPELIST_CHAIN4(X0, X1, X2, X3)`
- `#define _GLIBCXX_TYPELIST_CHAIN5(X0, X1, X2, X3, X4)`
- `#define _GLIBCXX_TYPELIST_CHAIN6(X0, X1, X2, X3, X4, X5)`
- `#define _GLIBCXX_TYPELIST_CHAIN7(X0, X1, X2, X3, X4, X5, X6)`
- `#define _GLIBCXX_TYPELIST_CHAIN8(X0, X1, X2, X3, X4, X5, X6, X7)`
- `#define _GLIBCXX_TYPELIST_CHAIN9(X0, X1, X2, X3, X4, X5, X6, X7, X8)`

### Functions

- `template<typename Fn, typename Typelist> void __gnu_cxx::typelist::apply (Fn &, Typelist)`
- `template<typename Gn, typename Typelist> void __gnu_cxx::typelist::apply_generator (Gn &, Typelist)`
- `template<typename Gn, typename TypelistT, typename TypelistV> void __gnu_cxx::typelist::apply_generator (Gn &, TypelistT, TypelistV)`
- `template<typename Fn, typename Typelist> void __gnu_cxx::typelist::apply_generator (Fn &fn, Typelist)`
- `template<typename Fn, typename TypelistT, typename TypelistV> void __gnu_cxx::typelist::apply_generator (Fn &fn, TypelistT, TypelistV)`

## 5.444.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Contains typelist\_chain definitions. Typelists are an idea by Andrei Alexandrescu.

## 5.445 types.h File Reference

## Namespaces

- [\\_\\_gnu\\_parallel](#)

## Typedefs

- typedef int64\_t [\\_\\_gnu\\_parallel::CASable](#)
- typedef uint64\_t [\\_\\_gnu\\_parallel::SequenceIndex](#)
- typedef uint16\_t [\\_\\_gnu\\_parallel::ThreadIndex](#)

## Enumerations

- enum [\\_\\_gnu\\_parallel::AlgorithmStrategy](#) { **heuristic**, **force\_sequential**, **force\_parallel** }
- enum [\\_\\_gnu\\_parallel::FindAlgorithm](#) { **GROWING\_BLOCKS**, **CONSTANT\_SIZE\_BLOCKS**, **EQUAL\_SPLIT** }
- enum [\\_\\_gnu\\_parallel::MultiwayMergeAlgorithm](#) { **LOSER\_TREE** }
- enum [\\_\\_gnu\\_parallel::Parallelism](#) { [\\_\\_gnu\\_parallel::sequential](#), [\\_\\_gnu\\_parallel::parallel\\_unbalanced](#), [\\_\\_gnu\\_parallel::parallel\\_balanced](#), [\\_\\_gnu\\_parallel::parallel\\_omp\\_loop](#), [\\_\\_gnu\\_parallel::parallel\\_omp\\_loop\\_static](#), [\\_\\_gnu\\_parallel::parallel\\_taskqueue](#) }
- enum [\\_\\_gnu\\_parallel::PartialSumAlgorithm](#) { **RECURSIVE**, **LINEAR** }
- enum [\\_\\_gnu\\_parallel::SortAlgorithm](#) { **MWMS**, **QS**, **QS\_BALANCED** }
- enum [\\_\\_gnu\\_parallel::SplittingAlgorithm](#) { **SAMPLING**, **EXACT** }

## Variables

- static const int [\\_\\_gnu\\_parallel::CASable\\_bits](#)
- static const [CASable](#) [\\_\\_gnu\\_parallel::CASable\\_mask](#)

## 5.445.1 Detailed Description

Basic types and typedefs. This file is a GNU parallel extension to the Standard C++ Library.

## 5.446 types\_traits.hpp File Reference

## Classes

- struct [\\_\\_gnu\\_pbds::detail::no\\_throw\\_copies< Key, Mapped >](#)
- struct [\\_\\_gnu\\_pbds::detail::no\\_throw\\_copies< Key, null\\_type >](#)
- struct [\\_\\_gnu\\_pbds::detail::stored\\_data< \\_Tv, \\_Th >](#)
- struct [\\_\\_gnu\\_pbds::detail::stored\\_data< \\_Tv, null\\_type >](#)
- struct [\\_\\_gnu\\_pbds::detail::stored\\_hash< \\_Th >](#)
- struct [\\_\\_gnu\\_pbds::detail::stored\\_value< \\_Tv >](#)

- struct [\\_\\_gnu\\_pbds::detail::type\\_base< Key, Mapped, \\_Alloc, Store\\_Hash >](#)
- struct [\\_\\_gnu\\_pbds::detail::type\\_base< Key, Mapped, \\_Alloc, false >](#)
- struct [\\_\\_gnu\\_pbds::detail::type\\_base< Key, Mapped, \\_Alloc, true >](#)
- struct [\\_\\_gnu\\_pbds::detail::type\\_base< Key, null\\_type, \\_Alloc, false >](#)
- struct [\\_\\_gnu\\_pbds::detail::type\\_base< Key, null\\_type, \\_Alloc, true >](#)
- struct [\\_\\_gnu\\_pbds::detail::type\\_dispatch< Key, Mapped, \\_Alloc, Store\\_Hash >](#)
- struct [\\_\\_gnu\\_pbds::detail::types\\_traits< Key, Mapped, \\_Alloc, Store\\_Hash >](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.446.1 Detailed Description

Contains a traits class of types used by containers.

#### 5.447 [unique\\_copy.h](#) File Reference

#### Namespaces

- [\\_\\_gnu\\_parallel](#)

#### Functions

- template<typename [\\_Iter](#) , class [\\_OutputIterator](#) , class [\\_BinaryPredicate](#) > [\\_OutputIterator \\_\\_gnu\\_parallel::\\_\\_parallel\\_unique\\_copy](#) ([\\_Iter](#) \_\_first, [\\_Iter](#) \_\_last, [\\_OutputIterator](#) \_\_result, [\\_BinaryPredicate](#) \_\_binary\_pred)
- template<typename [\\_Iter](#) , class [\\_OutputIterator](#) > [\\_OutputIterator \\_\\_gnu\\_parallel::\\_\\_parallel\\_unique\\_copy](#) ([\\_Iter](#) \_\_first, [\\_Iter](#) \_\_last, [\\_OutputIterator](#) \_\_result)

#### 5.447.1 Detailed Description

Parallel implementations of `std::unique_copy()`. This file is a GNU parallel extension to the Standard C++ Library.

#### 5.448 [unique\\_ptr.h](#) File Reference

#### Classes

- struct [std::default\\_delete< \\_Tp >](#)
- struct [std::default\\_delete< \\_Tp\[\] >](#)
- struct [std::hash< \[unique\\\_ptr\]\(#\)< \\_Tp, \\_Dp > >](#)
- class [std::unique\\_ptr< \\_Tp, \\_Dp >](#)
- class [std::unique\\_ptr< \\_Tp\[\], \\_Dp >](#)

#### Namespaces

- [std](#)

## Functions

- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep> bool std::operator!= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp> bool std::operator!= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<typename _Tp, typename _Dp> bool std::operator!= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep> bool std::operator< (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp> bool std::operator< (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp> bool std::operator< (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep> bool std::operator<= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp> bool std::operator<= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp> bool std::operator<= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep> bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp> bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<typename _Tp, typename _Dp> bool std::operator== (nullptr_t, const unique_ptr< _Tp, _Dp > &__x) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep> bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp> bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp> bool std::operator> (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep> bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp> bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp> bool std::operator>= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp> void std::swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y) noexcept`

## 5.448.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 5.449 unordered\_base.h File Reference

## Namespaces

- [std](#)
- [std::\\_\\_profile](#)

## Functions

- `template<typename _UnorderedCont, typename _Value, bool _Cache_hash_code> bool std::__profile::__are_equal (const _UnorderedCont &__uc, const __detail::__Hash_node< _Value, _Cache_hash_code > *__lhs, const __detail::__Hash_node< _Value, _Cache_hash_code > *__rhs)`
- `template<typename _UnorderedCont, typename _Value, bool _Cache_hash_code> std::size_t std::__profile::__get_bucket_index (const _UnorderedCont &__uc, const __detail::__Hash_node< _Value, _Cache_hash_code > *__node)`

### 5.449.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

## 5.450 unordered\_map.h File Reference

### Classes

- class `std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>`
- class `std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>`

### Namespaces

- `std`

### Typedefs

- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>, typename _Tr = __umap_traits<__cache_default<_Key, _Hash>::value>> using std::__umap_hashtable = _Hashtable<_Key, std::pair< const _Key, _Tp >, _Alloc, __detail::__Select1st, _Pred, _Hash, __detail::__Mod_range_hashing, __detail::__Default_ranged_hash, __detail::__Prime_rehash_policy, _Tr >`
- `template<bool _Cache> using std::__umap_traits = __detail::__Hashtable_traits<_Cache, false, true >`
- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>, typename _Tr = __ummap_traits<__cache_default<_Key, _Hash>::value>> using std::__ummap_hashtable = _Hashtable<_Key, std::pair< const _Key, _Tp >, _Alloc, __detail::__Select1st, _Pred, _Hash, __detail::__Mod_range_hashing, __detail::__Default_ranged_hash, __detail::__Prime_rehash_policy, _Tr >`
- `template<bool _Cache> using std::__ummap_traits = __detail::__Hashtable_traits<_Cache, false, false >`

### Functions

- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc> bool std::operator!= (const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc> bool std::operator!= (const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc> bool std::operator== (const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc> bool std::operator== (const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc> void std::swap (unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc> void std::swap (unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`

### 5.450.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_map>`.

## 5.451 unordered\_set.h File Reference

## Classes

- class `std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>`
- class `std::unordered_set<_Value, _Hash, _Pred, _Alloc>`

## Namespaces

- `std`

## Typedefs

- `template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __umset_traits<__cache_default<_Value, _Hash>::value>> using std::__umset ← hashtable = _Hashtable<_Value, _Value, _Alloc, __detail::_Identity, _Pred, _Hash, __detail::_Mod_range ← hashing, __detail::_Default_ranged_hash, __detail::_Prime_rehash_policy, _Tr>`
- `template<bool _Cache> using std::__umset_traits = __detail::_Hashtable_traits<_Cache, true, false>`
- `template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std ← ::allocator<_Value>, typename _Tr = __uset_traits<__cache_default<_Value, _Hash>::value>> using std::__uset_hashtable = _Hashtable<_Value, _Value, _Alloc, __detail::_Identity, _Pred, _Hash, __detail::_Mod_range_hashing, _ ← __detail::_Default_ranged_hash, __detail::_Prime_rehash_policy, _Tr>`
- `template<bool _Cache> using std::__uset_traits = __detail::_Hashtable_traits<_Cache, true, true>`

## Functions

- `template<class _Value, class _Hash, class _Pred, class _Alloc> bool std::operator!= (const unordered_set<_Value, _Hash, _Pred, _Alloc> &__x, const unordered_set<_Value, _Hash, _Pred, _Alloc> &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc> bool std::operator!= (const unordered_multiset<_Value, _Hash, _Pred, _Alloc> &__x, const unordered_multiset<_Value, _Hash, _Pred, _Alloc> &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc> bool std::operator== (const unordered_set<_Value, _Hash, _Pred, _Alloc> &__x, const unordered_set<_Value, _Hash, _Pred, _Alloc> &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc> bool std::operator== (const unordered_multiset<_Value, _Hash, _Pred, _Alloc> &__x, const unordered_multiset<_Value, _Hash, _Pred, _Alloc> &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc> void std::swap (unordered_set<_Value, _Hash, _Pred, _Alloc> &__x, unordered_set<_Value, _Hash, _Pred, _Alloc> &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc> void std::swap (unordered_multiset<_Value, _Hash, _Pred, _Alloc> &__x, unordered_multiset<_Value, _Hash, _Pred, _Alloc> &__y)`

## 5.451.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_set>`.

## 5.452 update\_fn\_imps.hpp File Reference

## 5.452.1 Detailed Description

Contains an implementation class for `pat_trie_`.



## 5.453 valarray\_after.h File Reference

### Namespaces

- [std](#)

### Macros

- `#define _DEFINE_EXPR_BINARY_FUNCTION(_Fun, _UFun)`
- `#define _DEFINE_EXPR_BINARY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_EXPR_UNARY_FUNCTION(_Name, _UName)`
- `#define _DEFINE_EXPR_UNARY_OPERATOR(_Op, _Name)`

### Functions

- `template<class _Dom > _Expr< _UnClos< _Abs, _Expr, _Dom >, typename _Dom::value_type > std::abs (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp > _Expr< _UnClos< _Abs, _ValArray, _Tp >, _Tp > std::abs (const valarray< _Tp > &__v)`
- `template<class _Dom > _Expr< _UnClos< _Acos, _Expr, _Dom >, typename _Dom::value_type > std::acos (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp > _Expr< _UnClos< _Acos, _ValArray, _Tp >, _Tp > std::acos (const valarray< _Tp > &__v)`
- `template<class _Dom > _Expr< _UnClos< _Asin, _Expr, _Dom >, typename _Dom::value_type > std::asin (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp > _Expr< _UnClos< _Asin, _ValArray, _Tp >, _Tp > std::asin (const valarray< _Tp > &__v)`
- `template<class _Dom > _Expr< _UnClos< _Atan, _Expr, _Dom >, typename _Dom::value_type > std::atan (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp > _Expr< _UnClos< _Atan, _ValArray, _Tp >, _Tp > std::atan (const valarray< _Tp > &__v)`
- `template<class _Dom1, class _Dom2 > _Expr< _BinClos< _Atan2, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > std::atan2 (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type > &__e2)`
- `template<class _Dom > _Expr< _BinClos< _Atan2, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom::value_type > std::atan2 (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom > _Expr< _BinClos< _Atan2, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > std::atan2 (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom > _Expr< _BinClos< _Atan2, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _Dom::value_type > std::atan2 (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom::value_type &__t)`
- `template<class _Dom > _Expr< _BinClos< _Atan2, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > std::atan2 (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp > _Expr< _BinClos< _Atan2, _ValArray, _ValArray, _Tp, _Tp >, _Tp > std::atan2 (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp > _Expr< _BinClos< _Atan2, _ValArray, _Constant, _Tp, _Tp >, _Tp > std::atan2 (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp > _Expr< _BinClos< _Atan2, _Constant, _ValArray, _Tp, _Tp >, _Tp > std::atan2 (const _Tp &__t, const valarray< _Tp > &__v)`

- `template<class _Dom > _Expr< _UnClos< _Cos, _Expr, _Dom >, typename _Dom::value_type > std::cos (const _Expr< _Dom, typename _Dom::value_type > &_e)`
- `template<typename _Tp > _Expr< _UnClos< _Cos, _ValArray, _Tp >, _Tp > std::cos (const valarray< _Tp > &_v)`
- `template<class _Dom > _Expr< _UnClos< _Cosh, _Expr, _Dom >, typename _Dom::value_type > std::cosh (const _Expr< _Dom, typename _Dom::value_type > &_e)`
- `template<typename _Tp > _Expr< _UnClos< _Cosh, _ValArray, _Tp >, _Tp > std::cosh (const valarray< _Tp > &_v)`
- `template<class _Dom > _Expr< _UnClos< _Exp, _Expr, _Dom >, typename _Dom::value_type > std::exp (const _Expr< _Dom, typename _Dom::value_type > &_e)`
- `template<typename _Tp > _Expr< _UnClos< _Exp, _ValArray, _Tp >, _Tp > std::exp (const valarray< _Tp > &_v)`
- `template<typename _Tp > _Expr< _UnClos< _Log, _ValArray, _Tp >, _Tp > std::log (const valarray< _Tp > &_v)`
- `template<class _Dom > _Expr< _UnClos< _Log, _Expr, _Dom >, typename _Dom::value_type > std::log (const _Expr< _Dom, typename _Dom::value_type > &_e)`
- `template<class _Dom > _Expr< _UnClos< _Log10, _Expr, _Dom >, typename _Dom::value_type > std::log10 (const _Expr< _Dom, typename _Dom::value_type > &_e)`
- `template<typename _Tp > _Expr< _UnClos< _Log10, _ValArray, _Tp >, _Tp > std::log10 (const valarray< _Tp > &_v)`
- `template<class _Dom > _Expr< _BinClos< __not_equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __not_equal_to, typename _Dom::value_type >::result_type > std::operator!= (const typename _Dom::value_type &_t, const _Expr< _Dom, typename _Dom::value_type > &_v)`
- `template<class _Dom > _Expr< _BinClos< __not_equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __not_equal_to, typename _Dom::value_type >::result_type > std::operator!= (const _Expr< _Dom, typename _Dom::value_type > &_e, const valarray< typename _Dom::value_type > &_v)`
- `template<class _Dom > _Expr< _BinClos< __not_equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __not_equal_to, typename _Dom::value_type >::result_type > std::operator!= (const valarray< typename _Dom::value_type > &_v, const _Expr< _Dom, typename _Dom::value_type > &_e)`
- `template<class _Dom1, class _Dom2 > _Expr< _BinClos< __not_equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __not_equal_to, typename _Dom1::value_type >::result_type > std::operator!= (const _Expr< _Dom1, typename _Dom1::value_type > &_v, const _Expr< _Dom2, typename _Dom2::value_type > &_w)`
- `template<class _Dom > _Expr< _BinClos< __not_equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __not_equal_to, typename _Dom::value_type >::result_type > std::operator!= (const _Expr< _Dom, typename _Dom::value_type > &_v, const typename _Dom::value_type &_t)`
- `template<class _Dom > _Expr< _BinClos< __modulus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __modulus, typename _Dom::value_type >::result_type > std::operator% (const _Expr< _Dom, typename _Dom::value_type > &_v, const typename _Dom::value_type &_t)`
- `template<class _Dom1, class _Dom2 > _Expr< _BinClos< __modulus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __modulus, typename _Dom1::value_type >::result_type > std::operator% (const _Expr< _Dom1, typename _Dom1::value_type > &_v, const _Expr< _Dom2, typename _Dom2::value_type > &_w)`
- `template<class _Dom > _Expr< _BinClos< __modulus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __modulus, typename _Dom::value_type >::result_type > std::operator% (const typename _Dom::value_type &_t, const _Expr< _Dom, typename _Dom::value_type > &_v)`
- `template<class _Dom > _Expr< _BinClos< __modulus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __modulus, typename _Dom::value_type >::result_type > std::operator% (const _Expr< _Dom, typename _Dom::value_type > &_e, const valarray< typename _Dom::value_type > &_v)`
- `template<class _Dom > _Expr< _BinClos< __modulus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __modulus, typename _Dom::value_type >::result_type > std::operator% (const valarray< typename _Dom::value_type > &_v, const _Expr< _Dom, typename _Dom::value_type > &_e)`
- `template<class _Dom > _Expr< _BinClos< __bitwise_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __bitwise_and, typename _Dom::value_type >::result_type > std::operator& (const _Expr< _Dom, typename _Dom::value_type > &_v, const typename _Dom::value_type &_t)`



- Generated on Tue Jun 30 2015 12:19:40 for libstdc++ by Doxygen



[illegible]

- `template<class _Dom > _Expr< _BinClos< __bitwise_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __bitwise_or, typename _Dom::value_type >::result_type > std::operator| (const typename _Dom::value_type & __t, const _Expr< _Dom, typename _Dom::value_type > & __v)`
- `template<class _Dom > _Expr< _BinClos< __bitwise_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __bitwise_or, typename _Dom::value_type >::result_type > std::operator| (const valarray< typename _Dom::value_type > & __v, const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<class _Dom > _Expr< _BinClos< __logical_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __logical_or, typename _Dom::value_type >::result_type > std::operator| (const valarray< typename _Dom::value_type > & __v, const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<class _Dom > _Expr< _BinClos< __logical_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __logical_or, typename _Dom::value_type >::result_type > std::operator| (const _Expr< _Dom, typename _Dom::value_type > & __v, const typename _Dom::value_type & __t)`
- `template<class _Dom > _Expr< _BinClos< __logical_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __logical_or, typename _Dom::value_type >::result_type > std::operator| (const _Expr< _Dom, typename _Dom::value_type > & __e, const valarray< typename _Dom::value_type > & __v)`
- `template<class _Dom > _Expr< _BinClos< __logical_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __logical_or, typename _Dom::value_type >::result_type > std::operator| (const typename _Dom::value_type & __t, const _Expr< _Dom, typename _Dom::value_type > & __v)`
- `template<class _Dom1, class _Dom2 > _Expr< _BinClos< __logical_or, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __logical_or, typename _Dom1::value_type >::result_type > std::operator| (const _Expr< _Dom1, typename _Dom1::value_type > & __v, const _Expr< _Dom2, typename _Dom2::value_type > & __w)`
- `template<class _Dom > _Expr< _BinClos< _Pow, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > std::pow (const valarray< typename _Dom::valarray > & __v, const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<class _Dom > _Expr< _BinClos< _Pow, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > std::pow (const typename _Dom::value_type & __t, const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<typename _Tp > _Expr< _BinClos< _Pow, _ValArray, _Constant, _Tp, _Tp >, _Tp > std::pow (const valarray< _Tp > & __v, const _Tp & __t)`
- `template<class _Dom > _Expr< _BinClos< _Pow, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _Dom::value_type > std::pow (const _Expr< _Dom, typename _Dom::value_type > & __e, const typename _Dom::value_type & __t)`
- `template<typename _Tp > _Expr< _BinClos< _Pow, _ValArray, _ValArray, _Tp, _Tp >, _Tp > std::pow (const valarray< _Tp > & __v, const valarray< _Tp > & __w)`
- `template<class _Dom > _Expr< _BinClos< _Pow, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom::value_type > std::pow (const _Expr< _Dom, typename _Dom::value_type > & __e, const valarray< typename _Dom::value_type > & __v)`
- `template<class _Dom1, class _Dom2 > _Expr< _BinClos< _Pow, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __logical_or, typename _Dom1::value_type >::result_type > std::pow (const _Expr< _Dom1, typename _Dom1::value_type > & __e1, const _Expr< _Dom2, typename _Dom2::value_type > & __e2)`
- `template<typename _Tp > _Expr< _BinClos< _Pow, _Constant, _ValArray, _Tp, _Tp >, _Tp > std::pow (const _Tp & __t, const valarray< _Tp > & __v)`
- `template<class _Dom > _Expr< _UnClos< _Sin, _Expr, _Dom >, typename _Dom::value_type > std::sin (const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<typename _Tp > _Expr< _UnClos< _Sin, _ValArray, _Tp >, _Tp > std::sin (const valarray< _Tp > & __v)`
- `template<class _Dom > _Expr< _UnClos< _Sinh, _Expr, _Dom >, typename _Dom::value_type > std::sinh (const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<typename _Tp > _Expr< _UnClos< _Sinh, _ValArray, _Tp >, _Tp > std::sinh (const valarray< _Tp > & __v)`
- `template<typename _Tp > _Expr< _UnClos< _Sqrt, _ValArray, _Tp >, _Tp > std::sqrt (const valarray< _Tp > & __v)`
- `template<class _Dom > _Expr< _UnClos< _Sqrt, _Expr, _Dom >, typename _Dom::value_type > std::sqrt (const _Expr< _Dom, typename _Dom::value_type > & __e)`

- `template<typename _Tp> _Expr< _UnClos< _Tan, _ValArray, _Tp>, _Tp> std::tan (const valarray< _Tp> &__v)`
- `template<class _Dom> _Expr< _UnClos< _Tan, _Expr, _Dom>, typename _Dom::value_type> std::tan (const _Expr< _Dom, typename _Dom::value_type> &__e)`
- `template<class _Dom> _Expr< _UnClos< _Tanh, _Expr, _Dom>, typename _Dom::value_type> std::tanh (const _Expr< _Dom, typename _Dom::value_type> &__e)`
- `template<typename _Tp> _Expr< _UnClos< _Tanh, _ValArray, _Tp>, _Tp> std::tanh (const valarray< _Tp> &__v)`

#### 5.453.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 5.454 valarray\_array.h File Reference

### Namespaces

- [std](#)

### Macros

- `#define _DEFINE_ARRAY_FUNCTION(_Op, _Name)`

### Functions

- `template<typename _Tp> void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b)`
- `template<typename _Tp> void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *↵__restrict __b)`
- `template<typename _Tp> void std::__valarray_copy (const _Tp *__restrict __a, _Tp *__restrict __b, size_t __n, size_t __s)`
- `template<typename _Tp> void std::__valarray_copy (const _Tp *__restrict __src, size_t __n, size_t __s1, _Tp *↵__restrict __dst, size_t __s2)`
- `template<typename _Tp> void std::__valarray_copy (const _Tp *__restrict __a, const size_t *↵__restrict __i, _Tp *__restrict __b, size_t __n)`
- `template<typename _Tp> void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b, const size_t *↵__restrict __i)`
- `template<typename _Tp> void std::__valarray_copy (const _Tp *__restrict __src, size_t __n, const size_t *↵__restrict __i, _Tp *__restrict __dst, const size_t *↵__restrict __j)`
- `template<typename _Tp> void std::__valarray_copy (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b)`
- `template<typename _Tp> void std::__valarray_copy (_Array< _Tp> __a, size_t __n, size_t __s, _Array< _Tp> __b)`
- `template<typename _Tp> void std::__valarray_copy (_Array< _Tp> __a, _Array< _Tp> __b, size_t __n, size_t __s)`
- `template<typename _Tp> void std::__valarray_copy (_Array< _Tp> __a, size_t __n, size_t __s1, _Array< _Tp> __b, size_t __s2)`
- `template<typename _Tp> void std::__valarray_copy (_Array< _Tp> __a, _Array< size_t> __i, _Array< _Tp> __b, size_t __n)`
- `template<typename _Tp> void std::__valarray_copy (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b, _Array< size_t> __i)`



- `template<typename _Tp> void std::__valarray_copy (_Array< _Tp> __src, size_t __n, _Array< size_t> __i, _Array< _Tp> __dst, _Array< size_t> __j)`
- `template<typename _Tp> void std::__valarray_copy_construct (const _Tp *__b, const _Tp *__e, _Tp *__restrict, ← __o)`
- `template<typename _Tp> void std::__valarray_copy_construct (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__restrict __o)`
- `template<typename _Tp> void std::__valarray_copy_construct (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp *__restrict __o, size_t __n)`
- `template<typename _Tp> void std::__valarray_copy_construct (_Array< _Tp> __a, _Array< size_t> __i, ← _Array< _Tp> __b, size_t __n)`
- `template<typename _Tp> void std::__valarray_copy_construct (_Array< _Tp> __a, size_t __n, size_t __s, ← _Array< _Tp> __b)`
- `template<typename _Tp> void std::__valarray_default_construct (_Tp *__b, _Tp *__e)`
- `template<typename _Tp> void std::__valarray_destroy_elements (_Tp *__b, _Tp *__e)`
- `template<typename _Tp> void std::__valarray_fill (_Tp *__restrict __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp> void std::__valarray_fill (_Tp *__restrict __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp> void std::__valarray_fill (_Tp *__restrict __a, const size_t *__restrict __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp> void std::__valarray_fill (_Array< _Tp> __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp> void std::__valarray_fill (_Array< _Tp> __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp> void std::__valarray_fill (_Array< _Tp> __a, _Array< size_t> __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp> void std::__valarray_fill_construct (_Tp *__b, _Tp *__e, const _Tp __t)`
- `void * std::__valarray_get_memory (size_t __n)`
- `template<typename _Tp> _Tp *__restrict std::__valarray_get_storage (size_t __n)`
- `template<typename _Ta> _Ta::value_type std::__valarray_max (const _Ta &__a)`
- `template<typename _Ta> _Ta::value_type std::__valarray_min (const _Ta &__a)`
- `template<typename _Tp> _Tp std::__valarray_product (const _Tp *__f, const _Tp *__l)`
- `void std::__valarray_release_memory (void *__p)`
- `template<typename _Tp> _Tp std::__valarray_sum (const _Tp *__f, const _Tp *__l)`
- `template<typename _Tp, class _Dom> void std::__Array_augmented__bitwise_and (_Array< _Tp> __a, _Array< size_t> __i, const _Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::__Array_augmented__bitwise_and (_Array< _Tp> __a, _Array< bool> __m, ← _Array< _Tp> __b, size_t __n)`
- `template<typename _Tp> void std::__Array_augmented__bitwise_and (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b, _Array< bool> __m)`
- `template<typename _Tp, class _Dom> void std::__Array_augmented__bitwise_and (_Array< _Tp> __a, _Array< bool> __m, const _Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::__Array_augmented__bitwise_and (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b, _Array< size_t> __i)`
- `template<typename _Tp> void std::__Array_augmented__bitwise_and (_Array< _Tp> __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp> void std::__Array_augmented__bitwise_and (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b)`
- `template<typename _Tp, class _Dom> void std::__Array_augmented__bitwise_and (_Array< _Tp> __a, const _Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::__Array_augmented__bitwise_and (_Array< _Tp> __a, size_t __n, size_t __s, _Array< _Tp> __b)`
- `template<typename _Tp> void std::__Array_augmented__bitwise_and (_Array< _Tp> __a, _Array< _Tp> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom> void std::__Array_augmented__bitwise_and (_Array< _Tp> __a, size_t __s, const _Expr< _Dom, _Tp> &__e, size_t __n)`

- `template<typename _Tp> void std::Array_augmented___bitwise_and (_Array< _Tp> __a, _Array< size_t> __i, _Array< _Tp> __b, size_t __n)`
- `template<typename _Tp, class _Dom> void std::Array_augmented___bitwise_or (_Array< _Tp> __a, _Array< bool> __m, const _Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___bitwise_or (_Array< _Tp> __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp> void std::Array_augmented___bitwise_or (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b)`
- `template<typename _Tp, class _Dom> void std::Array_augmented___bitwise_or (_Array< _Tp> __a, const _Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___bitwise_or (_Array< _Tp> __a, size_t __n, size_t __s, _Array< _Tp> __b)`
- `template<typename _Tp> void std::Array_augmented___bitwise_or (_Array< _Tp> __a, _Array< _Tp> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom> void std::Array_augmented___bitwise_or (_Array< _Tp> __a, size_t __s, const _Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___bitwise_or (_Array< _Tp> __a, _Array< size_t> __i, _Array< _Tp> __b, size_t __n)`
- `template<typename _Tp, class _Dom> void std::Array_augmented___bitwise_or (_Array< _Tp> __a, _Array< size_t> __i, const _Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___bitwise_or (_Array< _Tp> __a, _Array< bool> __m, _Array< _Tp> __b, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___bitwise_or (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b, _Array< bool> __m)`
- `template<typename _Tp> void std::Array_augmented___bitwise_or (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b, _Array< size_t> __i)`
- `template<typename _Tp> void std::Array_augmented___bitwise_xor (_Array< _Tp> __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp> void std::Array_augmented___bitwise_xor (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b)`
- `template<typename _Tp, class _Dom> void std::Array_augmented___bitwise_xor (_Array< _Tp> __a, const _Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___bitwise_xor (_Array< _Tp> __a, size_t __n, size_t __s, _Array< _Tp> __b)`
- `template<typename _Tp> void std::Array_augmented___bitwise_xor (_Array< _Tp> __a, _Array< _Tp> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom> void std::Array_augmented___bitwise_xor (_Array< _Tp> __a, size_t __s, const _Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___bitwise_xor (_Array< _Tp> __a, _Array< size_t> __i, _Array< _Tp> __b, size_t __n)`
- `template<typename _Tp, class _Dom> void std::Array_augmented___bitwise_xor (_Array< _Tp> __a, _Array< bool> __m, const _Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp, class _Dom> void std::Array_augmented___bitwise_xor (_Array< _Tp> __a, _Array< size_t> __i, const _Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___bitwise_xor (_Array< _Tp> __a, _Array< bool> __m, _Array< _Tp> __b, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___bitwise_xor (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b, _Array< bool> __m)`
- `template<typename _Tp> void std::Array_augmented___bitwise_xor (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b, _Array< size_t> __i)`
- `template<typename _Tp> void std::Array_augmented___divides (_Array< _Tp> __a, size_t __n, const _Tp &__t)`

- `template<typename _Tp> void std::Array_augmented___divides (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b)`
- `template<typename _Tp, class _Dom> void std::Array_augmented___divides (_Array< _Tp> __a, const _Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___divides (_Array< _Tp> __a, size_t __n, size_t __s, _Array< _Tp> __b)`
- `template<typename _Tp> void std::Array_augmented___divides (_Array< _Tp> __a, _Array< _Tp> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom> void std::Array_augmented___divides (_Array< _Tp> __a, size_t __s, const _Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___divides (_Array< _Tp> __a, _Array< size_t> __i, _Array< _Tp> __b, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___divides (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b, _Array< size_t> __i)`
- `template<typename _Tp, class _Dom> void std::Array_augmented___divides (_Array< _Tp> __a, _Array< size_t> __i, const _Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___divides (_Array< _Tp> __a, _Array< bool> __m, _Array< _Tp> __b, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___divides (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b, _Array< bool> __m)`
- `template<typename _Tp, class _Dom> void std::Array_augmented___divides (_Array< _Tp> __a, _Array< bool> __m, const _Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp, class _Dom> void std::Array_augmented___minus (_Array< _Tp> __a, _Array< bool> __m, const _Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___minus (_Array< _Tp> __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp> void std::Array_augmented___minus (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b)`
- `template<typename _Tp> void std::Array_augmented___minus (_Array< _Tp> __a, _Array< size_t> __i, _Array< _Tp> __b, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___minus (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b, _Array< size_t> __i)`
- `template<typename _Tp, class _Dom> void std::Array_augmented___minus (_Array< _Tp> __a, size_t __s, const _Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp, class _Dom> void std::Array_augmented___minus (_Array< _Tp> __a, _Array< size_t> __i, const _Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___minus (_Array< _Tp> __a, _Array< _Tp> __b, size_t __n, size_t __s)`
- `template<typename _Tp> void std::Array_augmented___minus (_Array< _Tp> __a, size_t __n, size_t __s, _Array< _Tp> __b)`
- `template<typename _Tp, class _Dom> void std::Array_augmented___minus (_Array< _Tp> __a, const _Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___minus (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b, _Array< bool> __m)`
- `template<typename _Tp> void std::Array_augmented___minus (_Array< _Tp> __a, _Array< bool> __m, _Array< _Tp> __b, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___modulus (_Array< _Tp> __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp> void std::Array_augmented___modulus (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b)`
- `template<typename _Tp, class _Dom> void std::Array_augmented___modulus (_Array< _Tp> __a, const _Expr< _Dom, _Tp> &__e, size_t __n)`

- `template<typename _Tp> void std::Array_augmented___modulus (_Array< _Tp> __a, _Array< _Tp> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom> void std::Array_augmented___modulus (_Array< _Tp> __a, size_t __s, const Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___modulus (_Array< _Tp> __a, _Array< size_t> __i, _Array< _Tp> __b, size_t __n)`
- `template<typename _Tp, class _Dom> void std::Array_augmented___modulus (_Array< _Tp> __a, _Array< size_t> __i, const Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___modulus (_Array< _Tp> __a, _Array< bool> __m, _Array< _Tp> __b, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___modulus (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b, _Array< bool> __m)`
- `template<typename _Tp, class _Dom> void std::Array_augmented___modulus (_Array< _Tp> __a, _Array< bool> __m, const Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___modulus (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b, _Array< size_t> __i)`
- `template<typename _Tp> void std::Array_augmented___modulus (_Array< _Tp> __a, size_t __n, size_t __s, _Array< _Tp> __b)`
- `template<typename _Tp> void std::Array_augmented___multiplies (_Array< _Tp> __a, _Array< _Tp> __b, size_t __n, size_t __s)`
- `template<typename _Tp> void std::Array_augmented___multiplies (_Array< _Tp> __a, _Array< size_t> __i, _Array< _Tp> __b, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___multiplies (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b, _Array< size_t> __i)`
- `template<typename _Tp, class _Dom> void std::Array_augmented___multiplies (_Array< _Tp> __a, _Array< size_t> __i, const Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___multiplies (_Array< _Tp> __a, _Array< bool> __m, _Array< _Tp> __b, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___multiplies (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b, _Array< bool> __m)`
- `template<typename _Tp> void std::Array_augmented___multiplies (_Array< _Tp> __a, size_t __n, size_t __s, _Array< _Tp> __b)`
- `template<typename _Tp, class _Dom> void std::Array_augmented___multiplies (_Array< _Tp> __a, const Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp, class _Dom> void std::Array_augmented___multiplies (_Array< _Tp> __a, _Array< bool> __m, const Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp, class _Dom> void std::Array_augmented___multiplies (_Array< _Tp> __a, size_t __s, const Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___multiplies (_Array< _Tp> __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp> void std::Array_augmented___multiplies (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b)`
- `template<typename _Tp> void std::Array_augmented___plus (_Array< _Tp> __a, _Array< bool> __m, _Array< _Tp> __b, size_t __n)`
- `template<typename _Tp, class _Dom> void std::Array_augmented___plus (_Array< _Tp> __a, size_t __s, const Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::Array_augmented___plus (_Array< _Tp> __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp> void std::Array_augmented___plus (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b)`
- `template<typename _Tp> void std::Array_augmented___plus (_Array< _Tp> __a, _Array< _Tp> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom> void std::Array_augmented___plus (_Array< _Tp> __a, const Expr< _Dom, _Tp> &__e, size_t __n)`

- `template<typename _Tp> void std::Array_augmented_plus (Array<_Tp> __a, Array<size_t> __i, ↵  
Array<_Tp> __b, size_t __n)`
- `template<typename _Tp, class _Dom> void std::Array_augmented_plus (Array<_Tp> __a, Array<size_t>  
__i, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::Array_augmented_plus (Array<_Tp> __a, size_t __n, Array<_Tp>  
__b, Array<size_t> __i)`
- `template<typename _Tp> void std::Array_augmented_plus (Array<_Tp> __a, size_t __n, Array<_Tp>  
__b, Array<bool> __m)`
- `template<typename _Tp> void std::Array_augmented_plus (Array<_Tp> __a, size_t __n, size_t __s, ↵  
Array<_Tp> __b)`
- `template<typename _Tp, class _Dom> void std::Array_augmented_plus (Array<_Tp> __a, Array<bool>  
__m, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::Array_augmented_shift_left (Array<_Tp> __a, size_t __n, const _Tp  
&__t)`
- `template<typename _Tp, class _Dom> void std::Array_augmented_shift_left (Array<_Tp> __a, const Expr<  
_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp, class _Dom> void std::Array_augmented_shift_left (Array<_Tp> __a, size_t __s,  
const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::Array_augmented_shift_left (Array<_Tp> __a, Array<bool> __m,  
Array<_Tp> __b, size_t __n)`
- `template<typename _Tp> void std::Array_augmented_shift_left (Array<_Tp> __a, size_t __n, size_t __s,  
Array<_Tp> __b)`
- `template<typename _Tp> void std::Array_augmented_shift_left (Array<_Tp> __a, Array<size_t> __i,  
Array<_Tp> __b, size_t __n)`
- `template<typename _Tp> void std::Array_augmented_shift_left (Array<_Tp> __a, size_t __n, Array<_Tp  
> __b, Array<bool> __m)`
- `template<typename _Tp> void std::Array_augmented_shift_left (Array<_Tp> __a, Array<_Tp> __b,  
size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom> void std::Array_augmented_shift_left (Array<_Tp> __a, Array<  
size_t> __i, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::Array_augmented_shift_left (Array<_Tp> __a, size_t __n, Array<_Tp  
> __b)`
- `template<typename _Tp> void std::Array_augmented_shift_left (Array<_Tp> __a, size_t __n, Array<_Tp  
> __b, Array<size_t> __i)`
- `template<typename _Tp, class _Dom> void std::Array_augmented_shift_left (Array<_Tp> __a, Array<bool  
> __m, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::Array_augmented_shift_right (Array<_Tp> __a, Array<_Tp> __b,  
size_t __n, size_t __s)`
- `template<typename _Tp> void std::Array_augmented_shift_right (Array<_Tp> __a, size_t __n, size_t __s,  
Array<_Tp> __b)`
- `template<typename _Tp> void std::Array_augmented_shift_right (Array<_Tp> __a, Array<size_t> __i,  
Array<_Tp> __b, size_t __n)`
- `template<typename _Tp, class _Dom> void std::Array_augmented_shift_right (Array<_Tp> __a, Array<  
bool> __m, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp, class _Dom> void std::Array_augmented_shift_right (Array<_Tp> __a, const ↵  
Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::Array_augmented_shift_right (Array<_Tp> __a, Array<bool> __m,  
Array<_Tp> __b, size_t __n)`
- `template<typename _Tp> void std::Array_augmented_shift_right (Array<_Tp> __a, size_t __n, const _Tp  
&__t)`
- `template<typename _Tp> void std::Array_augmented_shift_right (Array<_Tp> __a, size_t __n, Array<  
_Tp> __b, Array<size_t> __i)`

- `template<typename _Tp, class _Dom> void std::Array_augmented__shift_right (_Array< _Tp> __a, size_t __s, const _Expr< _Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp> void std::Array_augmented__shift_right (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b)`
- `template<typename _Tp> void std::Array_augmented__shift_right (_Array< _Tp> __a, size_t __n, _Array< _Tp> __b, _Array< bool> __m)`
- `template<typename _Tp, class _Dom> void std::Array_augmented__shift_right (_Array< _Tp> __a, _Array< size_t> __i, const _Expr< _Dom, _Tp> &__e, size_t __n)`

#### 5.454.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

### 5.455 `valarray_before.h` File Reference

#### Namespaces

- [std](#)

#### 5.455.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

### 5.456 `vstring.h` File Reference

#### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base>](#)
- struct [std::hash< \\_\\_gnu\\_cxx::\\_\\_u16vstring>](#)
- struct [std::hash< \\_\\_gnu\\_cxx::\\_\\_u32vstring>](#)
- struct [std::hash< \\_\\_gnu\\_cxx::\\_\\_vstring>](#)
- struct [std::hash< \\_\\_gnu\\_cxx::\\_\\_wvstring>](#)

#### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

#### Functions

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename> class _Base> basic_istream< _CharT, _Traits> & std::getline (basic_istream< _CharT, _Traits> &__is, \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base> &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename> class _Base> basic_istream< _CharT, _Traits> & std::getline (basic_istream< _CharT, _Traits> &__is, \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base> &__str)`



- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool ↵ \_\_gnu\_cxx::operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool ↵ \_\_gnu\_cxx::operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool ↵ \_\_gnu\_cxx::operator<= (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool ↵ \_\_gnu\_cxx::operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, template< typename, typename, typename > class _Base> __enable_if< std::__is_char< _CharT >::__value, bool >::__type __gnu_cxx::operator== (const __versa_string< _CharT, std::char\_traits< _CharT >, std::allocator< _CharT >, _Base > &__lhs, const __versa_string< _CharT, std::char\_traits< _CharT >, std::allocator< _CharT >, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool ↵ \_\_gnu\_cxx::operator== (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool ↵ \_\_gnu\_cxx::operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool ↵ \_\_gnu\_cxx::operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool ↵ \_\_gnu\_cxx::operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool ↵ \_\_gnu\_cxx::operator> (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool ↵ \_\_gnu\_cxx::operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool ↵ \_\_gnu\_cxx::operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool ↵ \_\_gnu\_cxx::operator>= (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> basic_↵ istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, \_\_gnu\_cxx::\_\_versa\_↵ string< _CharT, _Traits, _Alloc, _Base > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void ↵ \_\_gnu\_cxx::swap (__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`

#### 5.456.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.457 `vstring_fwd.h` File Reference

### Classes

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



## Namespaces

- [\\_\\_gnu\\_cxx](#)

## Typedefs

- typedef \_\_versa\_string< char, [std::char\\_traits](#)< char >, [std::allocator](#)< char >, \_\_rc\_string\_base > **\_\_gnu\_cxx::\_\_rc\_string**
- typedef \_\_vstring **\_\_gnu\_cxx::\_\_sso\_string**
- typedef \_\_versa\_string< char16\_t, [std::char\\_traits](#)< char16\_t >, [std::allocator](#)< char16\_t >, \_\_rc\_string\_base > **\_\_gnu\_cxx::\_\_u16rc\_string**
- typedef \_\_u16vstring **\_\_gnu\_cxx::\_\_u16sso\_string**
- typedef \_\_versa\_string< char16\_t > **\_\_gnu\_cxx::\_\_u16vstring**
- typedef \_\_versa\_string< char32\_t, [std::char\\_traits](#)< char32\_t >, [std::allocator](#)< char32\_t >, \_\_rc\_string\_base > **\_\_gnu\_cxx::\_\_u32rc\_string**
- typedef \_\_u32vstring **\_\_gnu\_cxx::\_\_u32sso\_string**
- typedef \_\_versa\_string< char32\_t > **\_\_gnu\_cxx::\_\_u32vstring**
- typedef \_\_versa\_string< char > **\_\_gnu\_cxx::\_\_vstring**
- typedef \_\_versa\_string< wchar\_t, [std::char\\_traits](#)< wchar\_t >, [std::allocator](#)< wchar\_t >, \_\_rc\_string\_base > **\_\_gnu\_cxx::\_\_wrc\_string**
- typedef \_\_wvstring **\_\_gnu\_cxx::\_\_wsso\_string**
- typedef \_\_versa\_string< wchar\_t > **\_\_gnu\_cxx::\_\_wvstring**

## 5.457.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

## 5.458 vstring\_util.h File Reference

## Namespaces

- [\\_\\_gnu\\_cxx](#)

## 5.458.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

## 5.459 workstealing.h File Reference

## Classes

- struct [\\_\\_gnu\\_parallel::\\_Job](#)< [\\_DifferenceTp](#) >

## Namespaces

- [\\_\\_gnu\\_parallel](#)

## Macros

- `#define _GLIBCXX_JOB_VOLATILE`

## Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result> _Op __gnu_parallel::__for_↔  
each_template_random_access_workstealing(_RAIter __begin, _RAIter __end, _Op __op, _Fu &__f, _Red __r,  
_Result __base, _Result &__output, typename std::iterator_traits<_RAIter>::difference_type __bound)`

### 5.459.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of work-stealing.

Work stealing is described in

R. D. Blumofe and C. E. Leiserson. Scheduling multithreaded computations by work stealing. *Journal of the ACM*, 46(5):720–748, 1999.

This file is a GNU parallel extension to the Standard C++ Library.



## Index

- `_AlgorithmStrategy`
  - `__gnu_parallel`, [253](#)
- `_AnyMatcher`
  - Base and Implementation Classes, [23](#)
- `_AutomatonPtr`
  - Base and Implementation Classes, [22](#)
- `_BALLOC_ALIGN_BYTES`
  - `bitmap_allocator.h`, [1900](#)
- `_BinIndex`
  - `__gnu_parallel`, [253](#)
- `_Bit_scan_forward`
  - `__gnu_cxx`, [230](#)
- `_CASable`
  - `__gnu_parallel`, [253](#)
- `_CASable_bits`
  - `__gnu_parallel`, [292](#)
- `_CASable_mask`
  - `__gnu_parallel`, [292](#)
- `_Construct`
  - `std`, [363](#)
- `_DRandomShufflingGlobalData`
  - `__gnu_parallel::DRandomShufflingGlobalData`, [620](#)
- `_Destroy`
  - `std`, [363](#), [364](#)
- `_Distance_precision`
  - `__gnu_debug`, [243](#)
- `_FindAlgorithm`
  - `__gnu_parallel`, [254](#)
- `_GLIBCXX_ASSERTIONS`
  - `compiletime_settings.h`, [1911](#)
- `_GLIBCXX_BAL_QUICKSORT`
  - `features.h`, [1933](#)
- `_GLIBCXX_CALL`
  - `compiletime_settings.h`, [1911](#)
- `_GLIBCXX_DEBUG_VERIFY_AT`
  - `macros.h`, [1966](#)
- `_GLIBCXX_DEQUE_BUF_SIZE`
  - `stl_deque.h`, [2056](#)
- `_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`
  - `features.h`, [1933](#)
- `_GLIBCXX_FIND_EQUAL_SPLIT`
  - `features.h`, [1933](#)
- `_GLIBCXX_FIND_GROWING_BLOCKS`
  - `features.h`, [1933](#)
- `_GLIBCXX_MERGESORT`
  - `features.h`, [1933](#)
- `_GLIBCXX_PARALLEL_CONDITION`
  - `settings.h`, [2036](#)
- `_GLIBCXX_PARALLEL_LENGTH`
  - `multiway_merge.h`, [1978](#)
- `_GLIBCXX_PROFILE_DEFINE_UNINIT_DATA`
  - `__gnu_profile`, [299](#)
- `_GLIBCXX_QUICKSORT`
  - `features.h`, [1933](#)
- `_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1`
  - `compiletime_settings.h`, [1912](#)
- `_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB`
  - `compiletime_settings.h`, [1912](#)
- `_GLIBCXX_SCALE_DOWN_FPU`
  - `compiletime_settings.h`, [1912](#)
- `_GLIBCXX_TREE_DYNAMIC_BALANCING`
  - `features.h`, [1933](#)
- `_GLIBCXX_TREE_FULL_COPY`
  - `features.h`, [1934](#)
- `_GLIBCXX_TREE_INITIAL_SPLITTING`
  - `features.h`, [1934](#)
- `_GLIBCXX_VERBOSE_LEVEL`
  - `compiletime_settings.h`, [1912](#)
- `_GLIBCXX_VOLATILE`
  - `partition.h`, [1989](#)
  - `queue.h`, [2008](#)
- `_GuardedIterator`
  - `__gnu_parallel::GuardedIterator`, [626](#)
- `_LoserTreeBase`
  - `__gnu_parallel::LoserTreeBase`, [643](#)
- `_M_allocate_and_copy`
  - `std::detail::Nfa`, [1035](#)
  - `std::vector`, [1846](#)
- `_M_allocate_single_object`
  - `__gnu_cxx::bitmap_allocator`, [519](#)
- `_M_attach`
  - `__gnu_debug::Safe_iterator`, [551](#)
  - `__gnu_debug::Safe_iterator_base`, [558](#)
  - `__gnu_debug::Safe_local_iterator`, [562](#)
  - `__gnu_debug::Safe_local_iterator_base`, [569](#)
  - `__gnu_debug::Safe_sequence`, [572](#)
  - `__gnu_debug::Safe_sequence_base`, [575](#)
  - `__gnu_debug::Safe_unordered_container`, [578](#)
  - `__gnu_debug::Safe_unordered_container_base`, [582](#)
  - `std::debug::map`, [985](#)
  - `std::debug::multimap`, [990](#)
  - `std::debug::multiset`, [994](#)
  - `std::debug::set`, [998](#)
- `_M_attach_local`
  - `__gnu_debug::Safe_unordered_container`, [578](#)
  - `__gnu_debug::Safe_unordered_container_base`, [582](#)
- `_M_attach_local_single`
  - `__gnu_debug::Safe_unordered_container`, [578](#)
  - `__gnu_debug::Safe_unordered_container_base`, [582](#)

- `_M_attach_single`
  - `__gnu_debug::__Safe_iterator`, 551
  - `__gnu_debug::__Safe_iterator_base`, 558
  - `__gnu_debug::__Safe_local_iterator`, 563
  - `__gnu_debug::__Safe_local_iterator_base`, 569
  - `__gnu_debug::__Safe_sequence`, 572
  - `__gnu_debug::__Safe_sequence_base`, 575
  - `__gnu_debug::__Safe_unordered_container`, 578
  - `__gnu_debug::__Safe_unordered_container_base`, 582
  - `std::__debug::map`, 985
  - `std::__debug::multimap`, 990
  - `std::__debug::multiset`, 994
  - `std::__debug::set`, 998
- `_M_attached_to`
  - `__gnu_debug::__Safe_iterator`, 551
  - `__gnu_debug::__Safe_iterator_base`, 558
  - `__gnu_debug::__Safe_local_iterator`, 563
  - `__gnu_debug::__Safe_local_iterator_base`, 569
- `_M_before_dereferenceable`
  - `__gnu_debug::__Safe_iterator`, 551
- `_M_begin`
  - `__gnu_parallel::__Piece`, 658
- `_M_bin_proc`
  - `__gnu_parallel::__DRandomShufflingGlobalData`, 620
- `_M_bins_begin`
  - `__gnu_parallel::__DRSSorterPU`, 622
- `_M_can_compare`
  - `__gnu_debug::__Safe_iterator`, 551
  - `__gnu_debug::__Safe_iterator_base`, 558
  - `__gnu_debug::__Safe_local_iterator`, 563
  - `__gnu_debug::__Safe_local_iterator_base`, 569
- `_M_clear`
  - `__gnu_cxx::free_list`, 527
- `_M_comp`
  - `__gnu_parallel::__LoserTree`, 637
  - `__gnu_parallel::__LoserTree< false, _Tp, _Compare >`, 641
  - `__gnu_parallel::__LoserTreeBase`, 644
- `_M_const_iterators`
  - `__gnu_debug::__Safe_sequence`, 573
  - `__gnu_debug::__Safe_sequence_base`, 576
  - `__gnu_debug::__Safe_unordered_container`, 580
  - `__gnu_debug::__Safe_unordered_container_base`, 584
  - `std::__debug::map`, 987
  - `std::__debug::multimap`, 991
  - `std::__debug::multiset`, 995
  - `std::__debug::set`, 999
- `_M_const_local_iterators`
  - `__gnu_debug::__Safe_unordered_container`, 580
  - `__gnu_debug::__Safe_unordered_container_base`, 584
- `_M_create_node`
  - `std::list`, 1438
- `_M_data`
  - `std::__List_node`, 1083
- `_M_deallocate_single_object`
  - `__gnu_cxx::bitmap_allocator`, 519
- `_M_dereferenceable`
  - `__gnu_debug::__Safe_iterator`, 552
  - `__gnu_debug::__Safe_local_iterator`, 563
- `_M_detach`
  - `__gnu_debug::__Safe_iterator`, 552
  - `__gnu_debug::__Safe_iterator_base`, 558
  - `__gnu_debug::__Safe_local_iterator`, 563
  - `__gnu_debug::__Safe_local_iterator_base`, 569
  - `__gnu_debug::__Safe_sequence`, 572
  - `__gnu_debug::__Safe_sequence_base`, 575
  - `__gnu_debug::__Safe_unordered_container`, 578
  - `__gnu_debug::__Safe_unordered_container_base`, 582
  - `std::__debug::map`, 985
  - `std::__debug::multimap`, 990
  - `std::__debug::multiset`, 994
  - `std::__debug::set`, 998
- `_M_detach_all`
  - `__gnu_debug::__Safe_sequence`, 572
  - `__gnu_debug::__Safe_sequence_base`, 575
  - `__gnu_debug::__Safe_unordered_container`, 578
  - `__gnu_debug::__Safe_unordered_container_base`, 583
  - `std::__debug::map`, 986
  - `std::__debug::multimap`, 990
  - `std::__debug::multiset`, 994
  - `std::__debug::set`, 998
- `_M_detach_local`
  - `__gnu_debug::__Safe_unordered_container`, 578
  - `__gnu_debug::__Safe_unordered_container_base`, 583
- `_M_detach_local_single`
  - `__gnu_debug::__Safe_unordered_container`, 579
  - `__gnu_debug::__Safe_unordered_container_base`, 583
- `_M_detach_single`
  - `__gnu_debug::__Safe_iterator`, 552
  - `__gnu_debug::__Safe_iterator_base`, 558
  - `__gnu_debug::__Safe_local_iterator`, 563
  - `__gnu_debug::__Safe_local_iterator_base`, 569
  - `__gnu_debug::__Safe_sequence`, 572
  - `__gnu_debug::__Safe_sequence_base`, 575
  - `__gnu_debug::__Safe_unordered_container`, 579
  - `__gnu_debug::__Safe_unordered_container_base`, 583
  - `std::__debug::map`, 986
  - `std::__debug::multimap`, 990
  - `std::__debug::multiset`, 994
  - `std::__debug::set`, 998

- `_M_detach_singular`
  - `__gnu_debug::__Safe_sequence`, 572
  - `__gnu_debug::__Safe_sequence_base`, 575
  - `__gnu_debug::__Safe_unordered_container`, 579
  - `__gnu_debug::__Safe_unordered_container_base`, 583
  - `std::__debug::map`, 986
  - `std::__debug::multimap`, 990
  - `std::__debug::multiset`, 994
  - `std::__debug::set`, 998
- `_M_dist`
  - `__gnu_parallel::__DRandomShufflingGlobalData`, 620
- `_M_elements_leftover`
  - `__gnu_parallel::__QSBThreadLocal`, 664
- `_M_end`
  - `__gnu_parallel::__Piece`, 658
- `_M_fill_initialize`
  - `std::deque`, 1309
- `_M_finish_iterator`
  - `__gnu_parallel::__accumulate_selector`, 586
  - `__gnu_parallel::__adjacent_difference_selector`, 586
  - `__gnu_parallel::__count_if_selector`, 592
  - `__gnu_parallel::__count_selector`, 593
  - `__gnu_parallel::__fill_selector`, 594
  - `__gnu_parallel::__for_each_selector`, 598
  - `__gnu_parallel::__generate_selector`, 601
  - `__gnu_parallel::__generic_for_each_selector`, 603
  - `__gnu_parallel::__identity_selector`, 604
  - `__gnu_parallel::__inner_product_selector`, 606
  - `__gnu_parallel::__replace_if_selector`, 614
  - `__gnu_parallel::__replace_selector`, 615
  - `__gnu_parallel::__transform1_selector`, 617
  - `__gnu_parallel::__transform2_selector`, 618
- `_M_first`
  - `__gnu_parallel::__Job`, 631
- `_M_first_insert`
  - `__gnu_parallel::__LoserTree`, 637
  - `__gnu_parallel::__LoserTree< false, _Tp, _Compare >`, 641
  - `__gnu_parallel::__LoserTreeBase`, 644
- `_M_get`
  - `__gnu_cxx::free_list`, 527
- `_M_get_mutex`
  - `__gnu_debug::__Safe_iterator`, 552
  - `__gnu_debug::__Safe_iterator_base`, 558
  - `__gnu_debug::__Safe_local_iterator`, 563
  - `__gnu_debug::__Safe_local_iterator_base`, 569
  - `__gnu_debug::__Safe_sequence`, 572
  - `__gnu_debug::__Safe_sequence_base`, 575
  - `__gnu_debug::__Safe_unordered_container`, 579
  - `__gnu_debug::__Safe_unordered_container_base`, 583
  - `std::__debug::map`, 986
  - `std::__debug::multimap`, 990
  - `std::__debug::multiset`, 995
  - `std::__debug::set`, 999
- `_M_getloc`
  - `std::basic_ios`, 1109
  - `std::ios_base`, 1411
- `_M_global`
  - `__gnu_parallel::__QSBThreadLocal`, 664
- `_M_in_same_bucket`
  - `__gnu_debug::__Safe_local_iterator`, 563
- `_M_incrementable`
  - `__gnu_debug::__Safe_iterator`, 552
  - `__gnu_debug::__Safe_local_iterator`, 564
- `_M_initial`
  - `__gnu_parallel::__QSBThreadLocal`, 664
- `_M_initialize_map`
  - `std::Deque_base`, 1068
  - `std::deque`, 1310
- `_M_insert`
  - `__gnu_cxx::free_list`, 527
- `_M_invalidate`
  - `__gnu_debug::__Safe_iterator`, 552
  - `__gnu_debug::__Safe_iterator_base`, 558
  - `__gnu_debug::__Safe_local_iterator`, 564
  - `__gnu_debug::__Safe_local_iterator_base`, 569
- `_M_invalidate_all`
  - `__gnu_debug::__Safe_sequence`, 572
  - `__gnu_debug::__Safe_sequence_base`, 576
  - `__gnu_debug::__Safe_unordered_container`, 579
  - `__gnu_debug::__Safe_unordered_container_base`, 583
  - `std::__debug::map`, 986
  - `std::__debug::multimap`, 990
  - `std::__debug::multiset`, 995
  - `std::__debug::set`, 999
- `_M_invalidate_if`
  - `__gnu_debug::__Safe_sequence`, 572
  - `__gnu_debug::__Safe_unordered_container`, 579
  - `std::__debug::map`, 986
  - `std::__debug::multimap`, 990
  - `std::__debug::multiset`, 995
  - `std::__debug::set`, 999
- `_M_invalidate_local_if`
  - `__gnu_debug::__Safe_unordered_container`, 579
- `_M_is_before_begin`
  - `__gnu_debug::__Safe_iterator`, 552
- `_M_is_begin`
  - `__gnu_debug::__Safe_iterator`, 552
  - `__gnu_debug::__Safe_local_iterator`, 564
- `_M_is_beginnest`
  - `__gnu_debug::__Safe_iterator`, 553
- `_M_is_end`
  - `__gnu_debug::__Safe_iterator`, 553
  - `__gnu_debug::__Safe_local_iterator`, 564
- `_M_iterators`

- \_\_gnu\_debug::\_\_Safe\_sequence, 573
  - \_\_gnu\_debug::\_\_Safe\_sequence\_base, 576
  - \_\_gnu\_debug::\_\_Safe\_unordered\_container, 580
  - \_\_gnu\_debug::\_\_Safe\_unordered\_container\_base, 584
- std::\_\_debug::map, 987
- std::\_\_debug::multimap, 991
- std::\_\_debug::multiset, 995
- std::\_\_debug::set, 999
- \_M\_key
  - \_\_gnu\_parallel::\_\_LoserTreeBase::\_\_Loser, 645
- \_M\_last
  - \_\_gnu\_parallel::\_\_Job, 631
- \_M\_leftover\_parts
  - \_\_gnu\_parallel::\_\_QSBThreadLocal, 664
- \_M\_load
  - \_\_gnu\_parallel::\_\_Job, 631
- \_M\_local\_iterators
  - \_\_gnu\_debug::\_\_Safe\_unordered\_container, 580
  - \_\_gnu\_debug::\_\_Safe\_unordered\_container\_base, 584
- \_M\_log\_k
  - \_\_gnu\_parallel::\_\_LoserTree, 637
  - \_\_gnu\_parallel::\_\_LoserTree< false, \_Tp, \_Compare >, 641
  - \_\_gnu\_parallel::\_\_LoserTreeBase, 644
- \_M\_losers
  - \_\_gnu\_parallel::\_\_LoserTree, 638
  - \_\_gnu\_parallel::\_\_LoserTree< false, \_Tp, \_Compare >, 641
  - \_\_gnu\_parallel::\_\_LoserTreeBase, 644
- \_M\_new\_elements\_at\_back
  - std::deque, 1310
- \_M\_new\_elements\_at\_front
  - std::deque, 1310
- \_M\_next
  - \_\_gnu\_debug::\_\_Safe\_iterator, 555
  - \_\_gnu\_debug::\_\_Safe\_iterator\_base, 559
  - \_\_gnu\_debug::\_\_Safe\_local\_iterator, 566
  - \_\_gnu\_debug::\_\_Safe\_local\_iterator\_base, 570
- \_M\_num\_bins
  - \_\_gnu\_parallel::\_\_DRandomShufflingGlobalData, 620
- \_M\_num\_bits
  - \_\_gnu\_parallel::\_\_DRandomShufflingGlobalData, 621
- \_M\_num\_threads
  - \_\_gnu\_parallel::\_\_DRSSorterPU, 622
  - \_\_gnu\_parallel::\_\_PMWMSSortingData, 660
  - \_\_gnu\_parallel::\_\_QSBThreadLocal, 664
- \_M\_offsets
  - \_\_gnu\_parallel::\_\_PMWMSSortingData, 660
- \_M\_pieces
  - \_\_gnu\_parallel::\_\_PMWMSSortingData, 660
- \_M\_pop\_back\_aux
  - std::deque, 1310
- \_M\_pop\_front\_aux
  - std::deque, 1310
- \_M\_prior
  - \_\_gnu\_debug::\_\_Safe\_iterator, 555
  - \_\_gnu\_debug::\_\_Safe\_iterator\_base, 559
  - \_\_gnu\_debug::\_\_Safe\_local\_iterator, 566
  - \_\_gnu\_debug::\_\_Safe\_local\_iterator\_base, 570
- \_M\_push\_back\_aux
  - std::deque, 1310
- \_M\_push\_front\_aux
  - std::deque, 1310
- \_M\_range\_check
  - std::\_\_detail::\_\_Nfa, 1035
  - std::deque, 1311
  - std::vector, 1846
- \_M\_range\_initialize
  - std::deque, 1311
- \_M\_reallocate\_map
  - std::deque, 1311
- \_M\_reserve\_elements\_at\_back
  - std::deque, 1312
- \_M\_reserve\_elements\_at\_front
  - std::deque, 1312
- \_M\_reserve\_map\_at\_back
  - std::deque, 1312
- \_M\_reserve\_map\_at\_front
  - std::deque, 1312
- \_M\_reset
  - \_\_gnu\_debug::\_\_Safe\_iterator, 553
  - \_\_gnu\_debug::\_\_Safe\_iterator\_base, 558
  - \_\_gnu\_debug::\_\_Safe\_local\_iterator, 564
  - \_\_gnu\_debug::\_\_Safe\_local\_iterator\_base, 569
- \_M\_revalidate\_singular
  - \_\_gnu\_debug::\_\_Safe\_sequence, 573
  - \_\_gnu\_debug::\_\_Safe\_sequence\_base, 576
  - \_\_gnu\_debug::\_\_Safe\_unordered\_container, 579
  - \_\_gnu\_debug::\_\_Safe\_unordered\_container\_base, 583
  - std::\_\_debug::map, 986
  - std::\_\_debug::multimap, 990
  - std::\_\_debug::multiset, 995
  - std::\_\_debug::set, 999
- \_M\_samples
  - \_\_gnu\_parallel::\_\_PMWMSSortingData, 660
- \_M\_sd
  - \_\_gnu\_parallel::\_\_DRSSorterPU, 622
- \_M\_seed
  - \_\_gnu\_parallel::\_\_DRSSorterPU, 622
- \_M\_sequence
  - \_\_gnu\_debug::\_\_Safe\_iterator, 556
  - \_\_gnu\_debug::\_\_Safe\_iterator\_base, 559
  - \_\_gnu\_debug::\_\_Safe\_local\_iterator, 566
  - \_\_gnu\_debug::\_\_Safe\_local\_iterator\_base, 570
- \_M\_sequential\_algorithm

- `__gnu_parallel::__adjacent_find_selector`, 587
  - `__gnu_parallel::__find_first_of_selector`, 596
  - `__gnu_parallel::__find_if_selector`, 597
  - `__gnu_parallel::__mismatch_selector`, 608
- `_M_set_node`
  - `std::Deque_iterator`, 1070
- `_M_singular`
  - `__gnu_debug::Safe_iterator`, 553
  - `__gnu_debug::Safe_iterator_base`, 559
  - `__gnu_debug::Safe_local_iterator`, 564
  - `__gnu_debug::Safe_local_iterator_base`, 569
- `_M_source`
  - `__gnu_parallel::DRandomShufflingGlobalData`, 621
  - `__gnu_parallel::LoserTreeBase::Loser`, 645
  - `__gnu_parallel::PMWMSSortingData`, 660
- `_M_starts`
  - `__gnu_parallel::DRandomShufflingGlobalData`, 621
  - `__gnu_parallel::PMWMSSortingData`, 660
- `_M_sup`
  - `__gnu_parallel::LoserTreeBase::Loser`, 645
- `_M_swap`
  - `__gnu_debug::Safe_sequence`, 573
  - `__gnu_debug::Safe_sequence_base`, 576
  - `__gnu_debug::Safe_unordered_container`, 579
  - `__gnu_debug::Safe_unordered_container_base`, 583
  - `std::debug::map`, 986
  - `std::debug::multimap`, 990
  - `std::debug::multiset`, 995
  - `std::debug::set`, 999
- `_M_temporaries`
  - `__gnu_parallel::DRandomShufflingGlobalData`, 621
- `_M_temporary`
  - `__gnu_parallel::PMWMSSortingData`, 661
- `_M_transfer_from_if`
  - `__gnu_debug::Safe_sequence`, 573
  - `std::debug::map`, 986
  - `std::debug::multimap`, 991
  - `std::debug::multiset`, 995
  - `std::debug::set`, 999
- `_M_unlink`
  - `__gnu_debug::Safe_iterator`, 553
  - `__gnu_debug::Safe_iterator_base`, 559
  - `__gnu_debug::Safe_local_iterator`, 564
  - `__gnu_debug::Safe_local_iterator_base`, 570
- `_M_use_pointer`
  - `__gnu_parallel::LoserTreeTraits`, 652
- `_M_version`
  - `__gnu_debug::Safe_iterator`, 556
  - `__gnu_debug::Safe_iterator_base`, 559
  - `__gnu_debug::Safe_local_iterator`, 566
  - `__gnu_debug::Safe_local_iterator_base`, 570
  - `__gnu_debug::Safe_sequence`, 573
  - `__gnu_debug::Safe_sequence_base`, 576
  - `__gnu_debug::Safe_unordered_container`, 580
  - `__gnu_debug::Safe_unordered_container_base`, 584
  - `std::debug::map`, 987
  - `std::debug::multimap`, 991
  - `std::debug::multiset`, 995
  - `std::debug::set`, 999
- `_Matcher`
  - Base and Implementation Classes, 22
- `_MultiwayMergeAlgorithm`
  - `__gnu_parallel`, 254
- `_Opcode`
  - Base and Implementation Classes, 23
- `_Parallelism`
  - `__gnu_parallel`, 254
- `_PartialSumAlgorithm`
  - `__gnu_parallel`, 254
- `_Piece`
  - `__gnu_parallel::QSBThreadLocal`, 663
- `_PseudoSequence`
  - `__gnu_parallel::PseudoSequence`, 661
- `_QSBThreadLocal`
  - `__gnu_parallel::QSBThreadLocal`, 663
- `_RandomNumber`
  - `__gnu_parallel::RandomNumber`, 665
- `_RestrictedBoundedConcurrentQueue`
  - `__gnu_parallel::RestrictedBoundedConcurrentQueue`, 666
- `_S_invalid_state_id`
  - Base and Implementation Classes, 23
- `_Safe_iterator`
  - `__gnu_debug::Safe_iterator`, 550
- `_Safe_iterator_base`
  - `__gnu_debug::Safe_iterator_base`, 557
- `_Safe_local_iterator`
  - `__gnu_debug::Safe_local_iterator`, 562
- `_Safe_local_iterator_base`
  - `__gnu_debug::Safe_local_iterator_base`, 568
- `_SequenceIndex`
  - `__gnu_parallel`, 253
- `_SortAlgorithm`
  - `__gnu_parallel`, 254
- `_SplittingAlgorithm`
  - `__gnu_parallel`, 254
- `_StatelDT`
  - Base and Implementation Classes, 22
- `_StateSet`
  - Base and Implementation Classes, 22
- `_StateStack`
  - Base and Implementation Classes, 23
- `_Tagger`
  - Base and Implementation Classes, 23
- `_Temporary_buffer`
  - `std::Temporary_buffer`, 1084



- `_ThreadIndex`
  - `__gnu_parallel`, 253
- `_TokenT`
  - Base and Implementation Classes, 23
- `__addressof`
  - Utilities, 219
- `__allocator_base`
  - Allocators, 8
- `__base`
  - `__gnu_debug`, 243
- `__begin1_iterator`
  - `__gnu_parallel::__inner_product_selector`, 606
- `__begin2_iterator`
  - `__gnu_parallel::__inner_product_selector`, 606
- `__bins_end`
  - `__gnu_parallel::DRSSorterPU`, 622
- `__bit_allocate`
  - `__gnu_cxx::__detail`, 239
- `__bit_free`
  - `__gnu_cxx::__detail`, 239
- `__calc_borders`
  - `__gnu_parallel`, 254
- `__check_dereferenceable`
  - `__gnu_debug`, 243
- `__check_singular`
  - `__gnu_debug`, 244
- `__check_singular_aux`
  - `__gnu_debug`, 244
- `__check_string`
  - `__gnu_debug`, 244
- `__compare_and_swap`
  - `__gnu_parallel`, 255
- `__ctype_type`
  - `std::basic_ios`, 1106
- `__cxxabiv1::__forced_unwind`, 432
- `__decode2`
  - `__gnu_parallel`, 255
- `__delete_min_insert`
  - `__gnu_parallel::LoserTree`, 637
  - `__gnu_parallel::LoserTree< false, _Tp, _Compare >`, 639
- `__determine_samples`
  - `__gnu_parallel`, 255
- `__encode2`
  - `__gnu_parallel`, 256
- `__env_t`
  - `__gnu_profile`, 298
- `__equally_split`
  - `__gnu_parallel`, 256
- `__equally_split_point`
  - `__gnu_parallel`, 256
- `__fetch_and_add`
  - `__gnu_parallel`, 257
- `__final_insertion_sort`
  - `std`, 354
- `__find`
  - `std`, 354, 355
- `__find_if`
  - `std`, 355
- `__find_if_not`
  - `std`, 355
- `__find_if_not_n`
  - `std`, 355
- `__find_template`
  - `__gnu_parallel`, 257, 259
- `__for_each_template_random_access`
  - `__gnu_parallel`, 260
- `__for_each_template_random_access_ed`
  - `__gnu_parallel`, 260
- `__for_each_template_random_access_omp_loop`
  - `__gnu_parallel`, 261
- `__for_each_template_random_access_omp_loop_static`
  - `__gnu_parallel`, 261
- `__for_each_template_random_access_workstealing`
  - `__gnu_parallel`, 262
- `__gcd`
  - `std`, 356
- `__genrand_bits`
  - `__gnu_parallel::RandomNumber`, 665
- `__get_distance`
  - `__gnu_debug`, 244
- `__get_min_source`
  - `__gnu_parallel::LoserTree`, 637
  - `__gnu_parallel::LoserTree< false, _Tp, _Compare >`, 639
  - `__gnu_parallel::LoserTreeBase`, 643
- `__get_num_threads`
  - `__gnu_parallel::balanced_quicksort_tag`, 676
  - `__gnu_parallel::balanced_tag`, 677
  - `__gnu_parallel::default_parallel_tag`, 678
  - `__gnu_parallel::exact_tag`, 680
  - `__gnu_parallel::multiway_mergesort_exact_tag`, 683
  - `__gnu_parallel::multiway_mergesort_sampling_tag`, 684
  - `__gnu_parallel::multiway_mergesort_tag`, 685
  - `__gnu_parallel::omp_loop_static_tag`, 687
  - `__gnu_parallel::omp_loop_tag`, 689
  - `__gnu_parallel::parallel_tag`, 693
  - `__gnu_parallel::quicksort_tag`, 694
  - `__gnu_parallel::sampling_tag`, 695
  - `__gnu_parallel::unbalanced_tag`, 698
- `__glibcxx_check_erase`
  - macros.h, 1965
- `__glibcxx_check_erase_after`
  - macros.h, 1965
- `__glibcxx_check_erase_range`
  - macros.h, 1965
- `__glibcxx_check_erase_range_after`

- macros.h, [1965](#)
- `__glibcxx_check_heap_pred`
  - macros.h, [1965](#)
- `__glibcxx_check_insert`
  - macros.h, [1965](#)
- `__glibcxx_check_insert_after`
  - macros.h, [1965](#)
- `__glibcxx_check_insert_range`
  - macros.h, [1965](#)
- `__glibcxx_check_insert_range_after`
  - macros.h, [1966](#)
- `__glibcxx_check_partitioned_lower`
  - macros.h, [1966](#)
- `__glibcxx_check_partitioned_lower_pred`
  - macros.h, [1966](#)
- `__glibcxx_check_partitioned_upper_pred`
  - macros.h, [1966](#)
- `__glibcxx_check_sorted_pred`
  - macros.h, [1966](#)
- `__gnu_cxx`, [223](#)
  - `_Bit_scan_forward`, [230](#)
  - `__static_pointer_cast`, [229](#)
  - `operator!=`, [230](#)
  - `operator<`, [233](#), [234](#)
  - `operator<=`, [234](#), [235](#)
  - `operator>`, [236](#), [237](#)
  - `operator>=`, [237](#), [238](#)
  - `operator+`, [231](#), [233](#)
  - `operator==`, [235](#), [236](#)
  - `swap`, [238](#)
- `__gnu_cxx::Caster<_ToType>`, [508](#)
- `__gnu_cxx::Char_types<_CharT>`, [509](#)
- `__gnu_cxx::ExtPtr_allocator<_Tp>`, [509](#)
- `__gnu_cxx::Invalid_type`, [510](#)
- `__gnu_cxx::Pointer_adapter<_Storage_policy>`, [510](#)
- `__gnu_cxx::Relative_pointer_impl<_Tp>`, [512](#)
- `__gnu_cxx::Relative_pointer_impl<const _Tp>`, [513](#)
- `__gnu_cxx::Std_pointer_impl<_Tp>`, [514](#)
- `__gnu_cxx::Unqualified_type<_Tp>`, [514](#)
- `__gnu_cxx::__alloc_traits`
  - `allocate`, [435](#)
  - `const_void_pointer`, [434](#)
  - `construct`, [435](#)
  - `deallocate`, [435](#)
  - `destroy`, [437](#)
  - `max_size`, [437](#)
  - `propagate_on_container_copy_assignment`, [434](#)
  - `propagate_on_container_move_assignment`, [434](#)
  - `propagate_on_container_swap`, [434](#)
  - `select_on_container_copy_construction`, [437](#)
  - `void_pointer`, [434](#)
- `__gnu_cxx::__alloc_traits<_Alloc>`, [433](#)
- `__gnu_cxx::__common_pool_policy<_PoolTp, _Thread>`, [438](#)
- `__gnu_cxx::__detail`, [238](#)
  - `__bit_allocate`, [239](#)
  - `__bit_free`, [239](#)
  - `__num_bitmaps`, [239](#)
  - `__num_blocks`, [239](#)
- `__gnu_cxx::__detail::Bitmap_counter<_Tp>`, [439](#)
- `__gnu_cxx::__detail::Ffit_finder`
  - `argument_type`, [440](#)
  - `result_type`, [440](#)
- `__gnu_cxx::__detail::Ffit_finder<_Tp>`, [440](#)
- `__gnu_cxx::__detail::__mini_vector<_Tp>`, [438](#)
- `__gnu_cxx::mt_alloc<_Tp, _Poolp>`, [441](#)
- `__gnu_cxx::mt_alloc_base<_Tp>`, [442](#)
- `__gnu_cxx::per_type_pool_policy<_Tp, _PoolTp, _Thread>`, [443](#)
- `__gnu_cxx::pool<_Thread>`, [443](#)
- `__gnu_cxx::pool<false>`, [444](#)
- `__gnu_cxx::pool<true>`, [445](#)
- `__gnu_cxx::pool_alloc<_Tp>`, [446](#)
- `__gnu_cxx::pool_alloc_base`, [447](#)
- `__gnu_cxx::pool_base`, [449](#)
- `__gnu_cxx::rc_string_base<_CharT, _Traits, _Alloc>`, [449](#)
- `__gnu_cxx::scoped_lock`, [451](#)
- `__gnu_cxx::versa_string`
  - `__versa_string`, [455–458](#)
  - `~__versa_string`, [458](#)
  - `append`, [458–460](#)
  - `assign`, [460](#), [461](#), [463](#), [465](#)
  - `at`, [465](#)
  - `back`, [467](#)
  - `begin`, [467](#)
  - `c_str`, [467](#)
  - `capacity`, [468](#)
  - `cbegin`, [468](#)
  - `cend`, [468](#)
  - `clear`, [468](#)
  - `compare`, [468–470](#)
  - `copy`, [471](#)
  - `crbegin`, [471](#)
  - `crend`, [471](#)
  - `data`, [472](#)
  - `empty`, [472](#)
  - `end`, [472](#)
  - `erase`, [472](#), [473](#)
  - `find`, [473](#), [474](#)
  - `find_first_not_of`, [476](#), [478](#)
  - `find_first_of`, [478](#), [480](#)
  - `find_last_not_of`, [480](#), [482](#)
  - `find_last_of`, [484](#), [486](#)
  - `front`, [486](#)
  - `get_allocator`, [486](#)
  - `insert`, [487–490](#)
  - `length`, [491](#)

- max\_size, 491
- npos, 508
- operator+=, 491, 493
- operator=, 493, 495
- operator[], 495
- pop\_back, 497
- push\_back, 497
- rbegin, 497
- rend, 498
- replace, 498, 500–504
- reserve, 504
- resize, 505
- rfind, 505, 506
- shrink\_to\_fit, 507
- size, 507
- substr, 507
- swap, 508
- \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_↵  
Base >, 452
- \_\_gnu\_cxx::annotate\_base, 515
- \_\_gnu\_cxx::array\_allocator< \_Tp, \_Array >, 516
- \_\_gnu\_cxx::array\_allocator\_base< \_Tp >, 517
- \_\_gnu\_cxx::bitmap\_allocator
  - \_M\_allocate\_single\_object, 519
  - \_M\_deallocate\_single\_object, 519
- \_\_gnu\_cxx::bitmap\_allocator< \_Tp >, 518
- \_\_gnu\_cxx::char\_traits< \_CharT >, 520
- \_\_gnu\_cxx::character< V, I, S >, 521
- \_\_gnu\_cxx::condition\_base, 522
- \_\_gnu\_cxx::debug\_allocator< \_Alloc >, 522
- \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 523
- \_\_gnu\_cxx::encoding\_char\_traits< \_CharT >, 524
- \_\_gnu\_cxx::encoding\_state, 525
- \_\_gnu\_cxx::forced\_error, 526
- \_\_gnu\_cxx::free\_list, 526
  - \_M\_clear, 527
  - \_M\_get, 527
  - \_M\_insert, 527
- \_\_gnu\_cxx::limit\_condition, 528
- \_\_gnu\_cxx::limit\_condition::always\_adjustor, 528
- \_\_gnu\_cxx::limit\_condition::limit\_adjustor, 529
- \_\_gnu\_cxx::limit\_condition::never\_adjustor, 529
- \_\_gnu\_cxx::malloc\_allocator< \_Tp >, 529
- \_\_gnu\_cxx::new\_allocator< \_Tp >, 530
- \_\_gnu\_cxx::random\_condition, 531
- \_\_gnu\_cxx::random\_condition::always\_adjustor, 532
- \_\_gnu\_cxx::random\_condition::group\_adjustor, 532
- \_\_gnu\_cxx::random\_condition::never\_adjustor, 533
- \_\_gnu\_cxx::recursive\_init\_error, 533
- \_\_gnu\_cxx::stdio\_filebuf
  - ~stdio\_filebuf, 535
  - fd, 535
  - file, 535
  - stdio\_filebuf, 534
- \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 533
- \_\_gnu\_cxx::stdio\_sync\_filebuf
  - file, 537
- \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 535
- \_\_gnu\_cxx::throw\_allocator\_base< \_Tp, \_Cond >, 537
- \_\_gnu\_cxx::throw\_allocator\_limit< \_Tp >, 539
- \_\_gnu\_cxx::throw\_allocator\_random< \_Tp >, 541
- \_\_gnu\_cxx::throw\_value\_base< \_Cond >, 542
- \_\_gnu\_cxx::throw\_value\_limit, 543
- \_\_gnu\_cxx::throw\_value\_random, 545
- \_\_gnu\_cxx::typelist, 239
  - apply\_generator, 240
- \_\_gnu\_debug, 240
  - \_Distance\_precision, 243
  - \_base, 243
  - \_check\_dereferenceable, 243
  - \_check\_singular, 244
  - \_check\_singular\_aux, 244
  - \_check\_string, 244
  - \_get\_distance, 244
  - \_valid\_range, 244, 245
  - \_valid\_range\_aux, 245
  - \_valid\_range\_aux2, 245
- \_\_gnu\_debug:: \_After\_nth\_from< \_Iterator >, 546
- \_\_gnu\_debug:: \_BeforeBeginHelper< \_Sequence >, 546
- \_\_gnu\_debug:: \_Equal\_to< \_Type >, 547
- \_\_gnu\_debug:: \_Not\_equal\_to< \_Type >, 547
- \_\_gnu\_debug:: \_Safe\_iterator
  - \_M\_attach, 551
  - \_M\_attach\_single, 551
  - \_M\_attached\_to, 551
  - \_M\_before\_dereferenceable, 551
  - \_M\_can\_compare, 551
  - \_M\_dereferenceable, 552
  - \_M\_detach, 552
  - \_M\_detach\_single, 552
  - \_M\_get\_mutex, 552
  - \_M\_incrementable, 552
  - \_M\_invalidate, 552
  - \_M\_is\_before\_begin, 552
  - \_M\_is\_begin, 552
  - \_M\_is\_beginnest, 553
  - \_M\_is\_end, 553
  - \_M\_next, 555
  - \_M\_prior, 555
  - \_M\_reset, 553
  - \_M\_sequence, 556
  - \_M\_singular, 553
  - \_M\_unlink, 553
  - \_M\_version, 556
  - \_Safe\_iterator, 550
  - base, 553
  - operator \_Iterator, 554
  - operator\*, 554

- operator++, 554
- operator->, 555
- operator--, 554
- operator=, 555
- \_\_gnu\_debug::\_\_Safe\_iterator< \_Iterator, \_Sequence >, 548
- \_\_gnu\_debug::\_\_Safe\_iterator\_base, 556
  - \_M\_attach, 558
  - \_M\_attach\_single, 558
  - \_M\_attached\_to, 558
  - \_M\_can\_compare, 558
  - \_M\_detach, 558
  - \_M\_detach\_single, 558
  - \_M\_get\_mutex, 558
  - \_M\_invalidate, 558
  - \_M\_next, 559
  - \_M\_prior, 559
  - \_M\_reset, 558
  - \_M\_sequence, 559
  - \_M\_singular, 559
  - \_M\_unlink, 559
  - \_M\_version, 559
  - \_Safe\_iterator\_base, 557
- \_\_gnu\_debug::\_\_Safe\_local\_iterator
  - \_M\_attach, 562
  - \_M\_attach\_single, 563
  - \_M\_attached\_to, 563
  - \_M\_can\_compare, 563
  - \_M\_dereferenceable, 563
  - \_M\_detach, 563
  - \_M\_detach\_single, 563
  - \_M\_get\_mutex, 563
  - \_M\_in\_same\_bucket, 563
  - \_M\_incrementable, 564
  - \_M\_invalidate, 564
  - \_M\_is\_begin, 564
  - \_M\_is\_end, 564
  - \_M\_next, 566
  - \_M\_prior, 566
  - \_M\_reset, 564
  - \_M\_sequence, 566
  - \_M\_singular, 564
  - \_M\_unlink, 564
  - \_M\_version, 566
  - \_Safe\_local\_iterator, 562
- base, 565
- bucket, 565
- operator \_Iterator, 565
- operator\*, 565
- operator++, 565
- operator->, 566
- operator=, 566
- \_\_gnu\_debug::\_\_Safe\_local\_iterator< \_Iterator, Sequence >, 560
- \_\_gnu\_debug::\_\_Safe\_local\_iterator\_base, 567
  - \_M\_attach, 569
  - \_M\_attach\_single, 569
  - \_M\_attached\_to, 569
  - \_M\_can\_compare, 569
  - \_M\_detach, 569
  - \_M\_detach\_single, 569
  - \_M\_get\_mutex, 569
  - \_M\_invalidate, 569
  - \_M\_next, 570
  - \_M\_prior, 570
  - \_M\_reset, 569
  - \_M\_sequence, 570
  - \_M\_singular, 569
  - \_M\_unlink, 570
  - \_M\_version, 570
  - \_Safe\_local\_iterator\_base, 568
- \_\_gnu\_debug::\_\_Safe\_sequence
  - \_M\_attach, 572
  - \_M\_attach\_single, 572
  - \_M\_const\_iterators, 573
  - \_M\_detach, 572
  - \_M\_detach\_all, 572
  - \_M\_detach\_single, 572
  - \_M\_detach\_singular, 572
  - \_M\_get\_mutex, 572
  - \_M\_invalidate\_all, 572
  - \_M\_invalidate\_if, 572
  - \_M\_iterators, 573
  - \_M\_revalidate\_singular, 573
  - \_M\_swap, 573
  - \_M\_transfer\_from\_if, 573
  - \_M\_version, 573
- \_\_gnu\_debug::\_\_Safe\_sequence< \_Sequence >, 571
- \_\_gnu\_debug::\_\_Safe\_sequence\_base, 574
  - \_M\_attach, 575
  - \_M\_attach\_single, 575
  - \_M\_const\_iterators, 576
  - \_M\_detach, 575
  - \_M\_detach\_all, 575
  - \_M\_detach\_single, 575
  - \_M\_detach\_singular, 575
  - \_M\_get\_mutex, 575
  - \_M\_invalidate\_all, 576
  - \_M\_iterators, 576
  - \_M\_revalidate\_singular, 576
  - \_M\_swap, 576
  - \_M\_version, 576
  - ~\_Safe\_sequence\_base, 575
- \_\_gnu\_debug::\_\_Safe\_unordered\_container
  - \_M\_attach, 578
  - \_M\_attach\_local, 578
  - \_M\_attach\_local\_single, 578
  - \_M\_attach\_single, 578

- [\\_M\\_const\\_iterators](#), 580
- [\\_M\\_const\\_local\\_iterators](#), 580
- [\\_M\\_detach](#), 578
- [\\_M\\_detach\\_all](#), 578
- [\\_M\\_detach\\_local](#), 578
- [\\_M\\_detach\\_local\\_single](#), 579
- [\\_M\\_detach\\_single](#), 579
- [\\_M\\_detach\\_singular](#), 579
- [\\_M\\_get\\_mutex](#), 579
- [\\_M\\_invalidate\\_all](#), 579
- [\\_M\\_invalidate\\_if](#), 579
- [\\_M\\_invalidate\\_local\\_if](#), 579
- [\\_M\\_iterators](#), 580
- [\\_M\\_local\\_iterators](#), 580
- [\\_M\\_revalidate\\_singular](#), 579
- [\\_M\\_swap](#), 579
- [\\_M\\_version](#), 580
- [\\_\\_gnu\\_debug:: Safe\\_unordered\\_container< \\_Container](#)  
    [>](#), 577
- [\\_\\_gnu\\_debug:: Safe\\_unordered\\_container\\_base](#), 581
  - [\\_M\\_attach](#), 582
  - [\\_M\\_attach\\_local](#), 582
  - [\\_M\\_attach\\_local\\_single](#), 582
  - [\\_M\\_attach\\_single](#), 582
  - [\\_M\\_const\\_iterators](#), 584
  - [\\_M\\_const\\_local\\_iterators](#), 584
  - [\\_M\\_detach](#), 582
  - [\\_M\\_detach\\_all](#), 583
  - [\\_M\\_detach\\_local](#), 583
  - [\\_M\\_detach\\_local\\_single](#), 583
  - [\\_M\\_detach\\_single](#), 583
  - [\\_M\\_detach\\_singular](#), 583
  - [\\_M\\_get\\_mutex](#), 583
  - [\\_M\\_invalidate\\_all](#), 583
  - [\\_M\\_iterators](#), 584
  - [\\_M\\_local\\_iterators](#), 584
  - [\\_M\\_revalidate\\_singular](#), 583
  - [\\_M\\_swap](#), 583
  - [\\_M\\_version](#), 584
  - [~\\_Safe\\_unordered\\_container\\_base](#), 582
- [\\_\\_gnu\\_internal](#), 245
- [\\_\\_gnu\\_parallel](#), 246
  - [\\_AlgorithmStrategy](#), 253
  - [\\_BinIndex](#), 253
  - [\\_CASable](#), 253
  - [\\_CASable\\_bits](#), 292
  - [\\_CASable\\_mask](#), 292
  - [\\_FindAlgorithm](#), 254
  - [\\_MultiwayMergeAlgorithm](#), 254
  - [\\_Parallelism](#), 254
  - [\\_PartialSumAlgorithm](#), 254
  - [\\_SequenceIndex](#), 253
  - [\\_SortAlgorithm](#), 254
  - [\\_SplittingAlgorithm](#), 254
  - [\\_ThreadIndex](#), 253
  - [\\_calc\\_borders](#), 254
  - [\\_compare\\_and\\_swap](#), 255
  - [\\_decode2](#), 255
  - [\\_determine\\_samples](#), 255
  - [\\_encode2](#), 256
  - [\\_equally\\_split](#), 256
  - [\\_equally\\_split\\_point](#), 256
  - [\\_fetch\\_and\\_add](#), 257
  - [\\_find\\_template](#), 257, 259
  - [\\_for\\_each\\_template\\_random\\_access](#), 260
  - [\\_for\\_each\\_template\\_random\\_access\\_ed](#), 260
  - [\\_for\\_each\\_template\\_random\\_access\\_omp\\_loop](#),  
    261
  - [\\_for\\_each\\_template\\_random\\_access\\_omp\\_loop↔](#)  
    [\\_static](#), 261
  - [\\_for\\_each\\_template\\_random\\_access\\_workstealing](#),  
    262
  - [\\_is\\_sorted](#), 262
  - [\\_median\\_of\\_three\\_iterators](#), 263
  - [\\_merge\\_advance](#), 263
  - [\\_merge\\_advance\\_movc](#), 263
  - [\\_merge\\_advance\\_usual](#), 264
  - [\\_parallel\\_merge\\_advance](#), 264, 265
  - [\\_parallel\\_nth\\_element](#), 265
  - [\\_parallel\\_partial\\_sort](#), 266
  - [\\_parallel\\_partial\\_sum](#), 266
  - [\\_parallel\\_partial\\_sum\\_basecase](#), 266
  - [\\_parallel\\_partial\\_sum\\_linear](#), 267
  - [\\_parallel\\_partition](#), 267
  - [\\_parallel\\_random\\_shuffle](#), 268
  - [\\_parallel\\_random\\_shuffle\\_drs](#), 268
  - [\\_parallel\\_random\\_shuffle\\_drs\\_pu](#), 268
  - [\\_parallel\\_sort](#), 269, 270, 272, 274
  - [\\_parallel\\_sort\\_qs](#), 275
  - [\\_parallel\\_sort\\_qs\\_conquer](#), 275
  - [\\_parallel\\_sort\\_qs\\_divide](#), 275
  - [\\_parallel\\_sort\\_qsb](#), 276
  - [\\_parallel\\_unique\\_copy](#), 276
  - [\\_qsb\\_conquer](#), 277
  - [\\_qsb\\_divide](#), 277
  - [\\_qsb\\_local\\_sort\\_with\\_helping](#), 277
  - [\\_random\\_number\\_pow2](#), 279
  - [\\_rd\\_log2](#), 279
  - [\\_round\\_up\\_to\\_pow2](#), 279
  - [\\_search\\_template](#), 279
  - [\\_sequential\\_multiway\\_merge](#), 281
  - [\\_sequential\\_random\\_shuffle](#), 281
  - [\\_shrink](#), 281
  - [\\_shrink\\_and\\_double](#), 282
  - [\\_yield](#), 282
- [list\\_partition](#), 282
- [max](#), 283
- [min](#), 283

- multiseq\_partition, 283
- multiseq\_selection, 283
- multiway\_merge, 284
- multiway\_merge\_3\_variant, 285
- multiway\_merge\_4\_variant, 286
- multiway\_merge\_exact\_splitting, 286
- multiway\_merge\_loser\_tree, 287
- multiway\_merge\_loser\_tree\_sentinel, 287
- multiway\_merge\_loser\_tree\_unguarded, 288
- multiway\_merge\_sampling\_splitting, 288
- multiway\_merge\_sentinels, 288
- parallel\_balanced, 254
- parallel\_multiway\_merge, 290
- parallel\_omp\_loop, 254
- parallel\_omp\_loop\_static, 254
- parallel\_sort\_mwms, 290
- parallel\_sort\_mwms\_pu, 292
- parallel\_taskqueue, 254
- parallel\_unbalanced, 254
- sequential, 254
- \_\_gnu\_parallel:: DRSSorterPU
  - \_M\_bins\_begin, 622
  - \_M\_num\_threads, 622
  - \_M\_sd, 622
  - \_M\_seed, 622
  - \_bins\_end, 622
- \_\_gnu\_parallel:: DRSSorterPU< \_RAIter, \_Random↵
  - NumberGenerator >, 621
- \_\_gnu\_parallel:: DRandomShufflingGlobalData
  - \_DRandomShufflingGlobalData, 620
  - \_M\_bin\_proc, 620
  - \_M\_dist, 620
  - \_M\_num\_bins, 620
  - \_M\_num\_bits, 621
  - \_M\_source, 621
  - \_M\_starts, 621
  - \_M\_temporaries, 621
- \_\_gnu\_parallel:: DRandomShufflingGlobalData< \_RAIter
  - >, 619
- \_\_gnu\_parallel:: DummyReduct, 622
- \_\_gnu\_parallel:: EqualFromLess
  - first\_argument\_type, 624
  - result\_type, 624
  - second\_argument\_type, 624
- \_\_gnu\_parallel:: EqualFromLess< \_T1, \_T2, \_Compare
  - >, 623
- \_\_gnu\_parallel:: EqualTo
  - first\_argument\_type, 625
  - result\_type, 625
  - second\_argument\_type, 625
- \_\_gnu\_parallel:: EqualTo< \_T1, \_T2 >, 624
- \_\_gnu\_parallel:: GuardedIterator
  - \_GuardedIterator, 626
  - operator \_RAIter, 626
- operator<, 627
- operator<=, 628
- operator\*, 626
- operator++, 626
- \_\_gnu\_parallel:: GuardedIterator< \_RAIter, \_Compare >,
  - 625
- \_\_gnu\_parallel:: IteratorPair
  - first, 629
  - second, 629
  - second\_type, 629
- \_\_gnu\_parallel:: IteratorPair< \_Iterator1, \_Iterator2, \_↵
  - IteratorCategory >, 628
- \_\_gnu\_parallel:: IteratorTriple< \_Iterator1, \_Iterator2, \_↵
  - Iterator3, \_IteratorCategory >, 630
- \_\_gnu\_parallel:: Job
  - \_M\_first, 631
  - \_M\_last, 631
  - \_M\_load, 631
- \_\_gnu\_parallel:: Job< \_DifferenceTp >, 631
- \_\_gnu\_parallel:: Less
  - first\_argument\_type, 632
  - result\_type, 632
  - second\_argument\_type, 632
- \_\_gnu\_parallel:: Less< \_T1, \_T2 >, 632
- \_\_gnu\_parallel:: Lexicographic
  - first\_argument\_type, 634
  - result\_type, 634
  - second\_argument\_type, 634
- \_\_gnu\_parallel:: Lexicographic< \_T1, \_T2, \_Compare >,
  - 633
- \_\_gnu\_parallel:: LexicographicReverse
  - first\_argument\_type, 635
  - result\_type, 635
  - second\_argument\_type, 635
- \_\_gnu\_parallel:: LexicographicReverse< \_T1, \_T2, \_↵
  - Compare >, 634
- \_\_gnu\_parallel:: LoserTree
  - \_M\_comp, 637
  - \_M\_first\_insert, 637
  - \_M\_log\_k, 637
  - \_M\_losers, 638
  - \_delete\_min\_insert, 637
  - \_get\_min\_source, 637
  - \_insert\_start, 637
- \_\_gnu\_parallel:: LoserTree< \_\_stable, \_Tp, \_Compare >,
  - 636
- \_\_gnu\_parallel:: LoserTree< false, \_Tp, \_Compare >,
  - 638
  - \_M\_comp, 641
  - \_M\_first\_insert, 641
  - \_M\_log\_k, 641
  - \_M\_losers, 641
  - \_delete\_min\_insert, 639
  - \_get\_min\_source, 639

- [\\_\\_init\\_winner](#), 639
- [\\_\\_insert\\_start](#), 639
- [\\_\\_gnu\\_parallel::\\_LoserTreeBase](#)
  - [\\_LoserTreeBase](#), 643
  - [\\_M\\_comp](#), 644
  - [\\_M\\_first\\_insert](#), 644
  - [\\_M\\_log\\_k](#), 644
  - [\\_M\\_losers](#), 644
  - [\\_get\\_min\\_source](#), 643
  - [\\_insert\\_start](#), 643
  - [~\\_LoserTreeBase](#), 643
- [\\_\\_gnu\\_parallel::\\_LoserTreeBase<\\_Tp, \\_Compare>](#), 642
- [\\_\\_gnu\\_parallel::\\_LoserTreeBase<\\_Tp, \\_Compare>::\\_Loser](#), 644
- [\\_\\_gnu\\_parallel::\\_LoserTreeBase::\\_Loser](#)
  - [\\_M\\_key](#), 645
  - [\\_M\\_source](#), 645
  - [\\_M\\_sup](#), 645
- [\\_\\_gnu\\_parallel::\\_LoserTreePointer<\\_\\_stable, \\_Tp, \\_Compare>](#), 646
- [\\_\\_gnu\\_parallel::\\_LoserTreePointer<false, \\_Tp, \\_Compare>](#), 647
- [\\_\\_gnu\\_parallel::\\_LoserTreePointerBase<\\_Tp, \\_Compare>](#), 648
- [\\_\\_gnu\\_parallel::\\_LoserTreePointerBase<\\_Tp, \\_Compare>::\\_Loser](#), 648
- [\\_\\_gnu\\_parallel::\\_LoserTreePointerUnguarded<\\_\\_stable, \\_Tp, \\_Compare>](#), 649
- [\\_\\_gnu\\_parallel::\\_LoserTreePointerUnguarded<false, \\_Tp, \\_Compare>](#), 650
- [\\_\\_gnu\\_parallel::\\_LoserTreePointerUnguardedBase<\\_Tp, \\_Compare>](#), 651
- [\\_\\_gnu\\_parallel::\\_LoserTreeTraits](#)
  - [\\_M\\_use\\_pointer](#), 652
- [\\_\\_gnu\\_parallel::\\_LoserTreeTraits<\\_Tp>](#), 652
- [\\_\\_gnu\\_parallel::\\_LoserTreeUnguarded<\\_\\_stable, \\_Tp, \\_Compare>](#), 653
- [\\_\\_gnu\\_parallel::\\_LoserTreeUnguarded<false, \\_Tp, \\_Compare>](#), 654
- [\\_\\_gnu\\_parallel::\\_LoserTreeUnguardedBase<\\_Tp, \\_Compare>](#), 655
- [\\_\\_gnu\\_parallel::Multiplies](#)
  - [first\\_argument\\_type](#), 656
  - [result\\_type](#), 656
  - [second\\_argument\\_type](#), 657
- [\\_\\_gnu\\_parallel::Multiplies<\\_Tp1, \\_Tp2, \\_Result>](#), 656
- [\\_\\_gnu\\_parallel::Nothing](#), 657
  - [operator\(\)](#), 657
- [\\_\\_gnu\\_parallel::PMWMSSortingData](#)
  - [\\_M\\_num\\_threads](#), 660
  - [\\_M\\_offsets](#), 660
  - [\\_M\\_pieces](#), 660
  - [\\_M\\_samples](#), 660
  - [\\_M\\_source](#), 660
  - [\\_M\\_starts](#), 660
  - [\\_M\\_temporary](#), 661
- [\\_\\_gnu\\_parallel::PMWMSSortingData<\\_RAIter>](#), 659
- [\\_\\_gnu\\_parallel::Piece](#)
  - [\\_M\\_begin](#), 658
  - [\\_M\\_end](#), 658
- [\\_\\_gnu\\_parallel::Piece<\\_DifferenceTp>](#), 657
- [\\_\\_gnu\\_parallel::Plus](#)
  - [first\\_argument\\_type](#), 659
  - [result\\_type](#), 659
  - [second\\_argument\\_type](#), 659
- [\\_\\_gnu\\_parallel::Plus<\\_Tp1, \\_Tp2, \\_Result>](#), 658
- [\\_\\_gnu\\_parallel::PseudoSequence](#)
  - [\\_PseudoSequence](#), 661
  - [begin](#), 662
  - [end](#), 662
- [\\_\\_gnu\\_parallel::PseudoSequence<\\_Tp, \\_DifferenceTp>](#), 661
- [\\_\\_gnu\\_parallel::PseudoSequenceIterator<\\_Tp, \\_DifferenceTp>](#), 662
- [\\_\\_gnu\\_parallel::QSBThreadLocal](#)
  - [\\_M\\_elements\\_leftover](#), 664
  - [\\_M\\_global](#), 664
  - [\\_M\\_initial](#), 664
  - [\\_M\\_leftover\\_parts](#), 664
  - [\\_M\\_num\\_threads](#), 664
  - [\\_Piece](#), 663
  - [\\_QSBThreadLocal](#), 663
- [\\_\\_gnu\\_parallel::QSBThreadLocal<\\_RAIter>](#), 663
- [\\_\\_gnu\\_parallel::RandomNumber](#), 664
  - [\\_RandomNumber](#), 665
  - [\\_genrand\\_bits](#), 665
  - [operator\(\)](#), 665
- [\\_\\_gnu\\_parallel::RestrictedBoundedConcurrentQueue](#)
  - [\\_RestrictedBoundedConcurrentQueue](#), 666
  - [~\\_RestrictedBoundedConcurrentQueue](#), 666
  - [pop\\_back](#), 666
  - [pop\\_front](#), 666
  - [push\\_front](#), 667
- [\\_\\_gnu\\_parallel::RestrictedBoundedConcurrentQueue<\\_Tp>](#), 666
- [\\_\\_gnu\\_parallel::SamplingSorter<\\_\\_stable, \\_RAIter, \\_StrictWeakOrdering>](#), 667
- [\\_\\_gnu\\_parallel::SamplingSorter<false, \\_RAIter, \\_StrictWeakOrdering>](#), 667
- [\\_\\_gnu\\_parallel::Settings](#), 668
  - [accumulate\\_minimal\\_n](#), 669
  - [adjacent\\_difference\\_minimal\\_n](#), 669
  - [cache\\_line\\_size](#), 669
  - [count\\_minimal\\_n](#), 670
  - [fill\\_minimal\\_n](#), 670
  - [find\\_increasing\\_factor](#), 670
  - [find\\_initial\\_block\\_size](#), 670
  - [find\\_maximum\\_block\\_size](#), 670



[find\\_scale\\_factor](#), 670  
[find\\_sequential\\_search\\_size](#), 670  
[for\\_each\\_minimal\\_n](#), 670  
[generate\\_minimal\\_n](#), 670  
[get](#), 669  
[L1\\_cache\\_size](#), 671  
[L2\\_cache\\_size](#), 671  
[max\\_element\\_minimal\\_n](#), 671  
[merge\\_minimal\\_n](#), 671  
[merge\\_oversampling](#), 671  
[min\\_element\\_minimal\\_n](#), 671  
[multiway\\_merge\\_minimal\\_k](#), 671  
[multiway\\_merge\\_minimal\\_n](#), 671  
[multiway\\_merge\\_oversampling](#), 671  
[nth\\_element\\_minimal\\_n](#), 671  
[partial\\_sort\\_minimal\\_n](#), 672  
[partial\\_sum\\_dilation](#), 672  
[partial\\_sum\\_minimal\\_n](#), 672  
[partition\\_chunk\\_share](#), 672  
[partition\\_chunk\\_size](#), 672  
[partition\\_minimal\\_n](#), 672  
[qsb\\_steals](#), 672  
[random\\_shuffle\\_minimal\\_n](#), 672  
[replace\\_minimal\\_n](#), 672  
[search\\_minimal\\_n](#), 673  
[set](#), 669  
[set\\_difference\\_minimal\\_n](#), 673  
[set\\_intersection\\_minimal\\_n](#), 673  
[set\\_symmetric\\_difference\\_minimal\\_n](#), 673  
[set\\_union\\_minimal\\_n](#), 673  
[sort\\_minimal\\_n](#), 673  
[sort\\_mwms\\_oversampling](#), 673  
[sort\\_qs\\_num\\_samples\\_preset](#), 673  
[sort\\_qsb\\_base\\_case\\_maximal\\_n](#), 673  
[TLB\\_size](#), 674  
[transform\\_minimal\\_n](#), 674  
[unique\\_copy\\_minimal\\_n](#), 674  
[\\_\\_gnu\\_parallel::SplitConsistently< \\_\\_exact, \\_RAIter, \\_Compare, \\_SortingPlacesIterator >](#), 674  
[\\_\\_gnu\\_parallel::SplitConsistently< false, \\_RAIter, \\_Compare, \\_SortingPlacesIterator >](#), 674  
[\\_\\_gnu\\_parallel::SplitConsistently< true, \\_RAIter, \\_Compare, \\_SortingPlacesIterator >](#), 675  
[\\_\\_gnu\\_parallel::accumulate\\_binop\\_reduct< \\_BinOp >](#), 584  
[\\_\\_gnu\\_parallel::accumulate\\_selector](#)  
[\\_M\\_finish\\_iterator](#), 586  
[operator\(\)](#), 585  
[\\_\\_gnu\\_parallel::accumulate\\_selector< \\_It >](#), 585  
[\\_\\_gnu\\_parallel::adjacent\\_difference\\_selector](#)  
[\\_M\\_finish\\_iterator](#), 586  
[\\_\\_gnu\\_parallel::adjacent\\_difference\\_selector< \\_It >](#), 586  
[\\_\\_gnu\\_parallel::adjacent\\_find\\_selector](#), 587  
[\\_M\\_sequential\\_algorithm](#), 587  
[operator\(\)](#), 588  
[\\_\\_gnu\\_parallel::binder1st](#)  
[argument\\_type](#), 589  
[result\\_type](#), 589  
[\\_\\_gnu\\_parallel::binder1st< \\_Operation, \\_FirstArgumentType, \\_SecondArgumentType, \\_ResultType >](#), 588  
[\\_\\_gnu\\_parallel::binder2nd](#)  
[argument\\_type](#), 591  
[result\\_type](#), 591  
[\\_\\_gnu\\_parallel::binder2nd< \\_Operation, \\_FirstArgumentType, \\_SecondArgumentType, \\_ResultType >](#), 590  
[\\_\\_gnu\\_parallel::count\\_if\\_selector](#)  
[\\_M\\_finish\\_iterator](#), 592  
[operator\(\)](#), 592  
[\\_\\_gnu\\_parallel::count\\_if\\_selector< \\_It, \\_Diff >](#), 591  
[\\_\\_gnu\\_parallel::count\\_selector](#)  
[\\_M\\_finish\\_iterator](#), 593  
[operator\(\)](#), 593  
[\\_\\_gnu\\_parallel::count\\_selector< \\_It, \\_Diff >](#), 592  
[\\_\\_gnu\\_parallel::fill\\_selector](#)  
[\\_M\\_finish\\_iterator](#), 594  
[operator\(\)](#), 594  
[\\_\\_gnu\\_parallel::fill\\_selector< \\_It >](#), 594  
[\\_\\_gnu\\_parallel::find\\_first\\_of\\_selector](#)  
[\\_M\\_sequential\\_algorithm](#), 596  
[operator\(\)](#), 596  
[\\_\\_gnu\\_parallel::find\\_first\\_of\\_selector< \\_FIterator >](#), 595  
[\\_\\_gnu\\_parallel::find\\_if\\_selector](#), 596  
[\\_M\\_sequential\\_algorithm](#), 597  
[operator\(\)](#), 597  
[\\_\\_gnu\\_parallel::for\\_each\\_selector](#)  
[\\_M\\_finish\\_iterator](#), 598  
[operator\(\)](#), 598  
[\\_\\_gnu\\_parallel::for\\_each\\_selector< \\_It >](#), 598  
[\\_\\_gnu\\_parallel::generate\\_selector](#)  
[\\_M\\_finish\\_iterator](#), 601  
[operator\(\)](#), 599  
[\\_\\_gnu\\_parallel::generate\\_selector< \\_It >](#), 599  
[\\_\\_gnu\\_parallel::generic\\_find\\_selector](#), 601  
[\\_\\_gnu\\_parallel::generic\\_for\\_each\\_selector](#)  
[\\_M\\_finish\\_iterator](#), 603  
[\\_\\_gnu\\_parallel::generic\\_for\\_each\\_selector< \\_It >](#), 602  
[\\_\\_gnu\\_parallel::identity\\_selector](#)  
[\\_M\\_finish\\_iterator](#), 604  
[operator\(\)](#), 604  
[\\_\\_gnu\\_parallel::identity\\_selector< \\_It >](#), 603  
[\\_\\_gnu\\_parallel::inner\\_product\\_selector](#)  
[\\_M\\_finish\\_iterator](#), 606  
[begin1\\_iterator](#), 606  
[begin2\\_iterator](#), 606



\_\_inner\_product\_selector, 605  
 operator(), 606  
 \_\_gnu\_parallel::\_\_inner\_product\_selector< \_It, \_It2, \_Tp  
 >, 605  
 \_\_gnu\_parallel::\_\_max\_element\_reduct< \_Compare, \_↵  
 It >, 607  
 \_\_gnu\_parallel::\_\_min\_element\_reduct< \_Compare, \_↵  
 It >, 607  
 \_\_gnu\_parallel::\_\_mismatch\_selector, 608  
 \_M\_sequential\_algorithm, 608  
 operator(), 608  
 \_\_gnu\_parallel::\_\_multiway\_merge\_3\_variant\_sentinel ↵  
 switch< \_\_sentinels, \_RAIterIterator, \_RAIter3,  
 \_DifferenceTp, \_Compare >, 609  
 \_\_gnu\_parallel::\_\_multiway\_merge\_3\_variant\_sentinel↵  
 \_switch< true, \_RAIterIterator, \_RAIter3, \_↵  
 DifferenceTp, \_Compare >, 609  
 \_\_gnu\_parallel::\_\_multiway\_merge\_4\_variant\_sentinel ↵  
 switch< \_\_sentinels, \_RAIterIterator, \_RAIter3,  
 \_DifferenceTp, \_Compare >, 610  
 \_\_gnu\_parallel::\_\_multiway\_merge\_4\_variant\_sentinel↵  
 \_switch< true, \_RAIterIterator, \_RAIter3, \_↵  
 DifferenceTp, \_Compare >, 610  
 \_\_gnu\_parallel::\_\_multiway\_merge\_k\_variant\_sentinel ↵  
 switch< \_\_sentinels, \_\_stable, \_RAIterIterator,  
 \_RAIter3, \_DifferenceTp, \_Compare >, 610  
 \_\_gnu\_parallel::\_\_multiway\_merge\_k\_variant\_sentinel ↵  
 switch< false, \_\_stable, \_RAIterIterator, \_RA↵  
 Iter3, \_DifferenceTp, \_Compare >, 611  
 \_\_gnu\_parallel::\_\_replace\_if\_selector  
 \_M\_finish\_iterator, 614  
 \_\_new\_val, 614  
 \_\_replace\_if\_selector, 612  
 operator(), 613  
 \_\_gnu\_parallel::\_\_replace\_if\_selector< \_It, \_Op, \_Tp >,  
 612  
 \_\_gnu\_parallel::\_\_replace\_selector  
 \_M\_finish\_iterator, 615  
 \_\_new\_val, 615  
 \_\_replace\_selector, 615  
 operator(), 615  
 \_\_gnu\_parallel::\_\_replace\_selector< \_It, \_Tp >, 614  
 \_\_gnu\_parallel::\_\_transform1\_selector  
 \_M\_finish\_iterator, 617  
 operator(), 616  
 \_\_gnu\_parallel::\_\_transform1\_selector< \_It >, 616  
 \_\_gnu\_parallel::\_\_transform2\_selector  
 \_M\_finish\_iterator, 618  
 operator(), 618  
 \_\_gnu\_parallel::\_\_transform2\_selector< \_It >, 617  
 \_\_gnu\_parallel::\_\_unary\_negate  
 argument\_type, 619  
 result\_type, 619  
 \_\_gnu\_parallel::\_\_unary\_negate< \_Predicate, argument↵  
 \_type >, 618  
 \_\_gnu\_parallel::balanced\_quicksort\_tag, 675  
 \_\_get\_num\_threads, 676  
 set\_num\_threads, 676  
 \_\_gnu\_parallel::balanced\_tag, 676  
 \_\_get\_num\_threads, 677  
 set\_num\_threads, 677  
 \_\_gnu\_parallel::constant\_size\_blocks\_tag, 677  
 \_\_gnu\_parallel::default\_parallel\_tag, 678  
 \_\_get\_num\_threads, 678  
 set\_num\_threads, 679  
 \_\_gnu\_parallel::equal\_split\_tag, 679  
 \_\_gnu\_parallel::exact\_tag, 680  
 \_\_get\_num\_threads, 680  
 set\_num\_threads, 680  
 \_\_gnu\_parallel::find\_tag, 681  
 \_\_gnu\_parallel::growing\_blocks\_tag, 682  
 \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag, 682  
 \_\_get\_num\_threads, 683  
 set\_num\_threads, 683  
 \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag, 684  
 \_\_get\_num\_threads, 684  
 set\_num\_threads, 684  
 \_\_gnu\_parallel::multiway\_mergesort\_tag, 685  
 \_\_get\_num\_threads, 685  
 set\_num\_threads, 685  
 \_\_gnu\_parallel::omp\_loop\_static\_tag, 687  
 \_\_get\_num\_threads, 687  
 set\_num\_threads, 687  
 \_\_gnu\_parallel::omp\_loop\_tag, 689  
 \_\_get\_num\_threads, 689  
 set\_num\_threads, 689  
 \_\_gnu\_parallel::parallel\_tag, 692  
 \_\_get\_num\_threads, 693  
 parallel\_tag, 693  
 set\_num\_threads, 693  
 \_\_gnu\_parallel::quicksort\_tag, 694  
 \_\_get\_num\_threads, 694  
 set\_num\_threads, 694  
 \_\_gnu\_parallel::sampling\_tag, 695  
 \_\_get\_num\_threads, 695  
 set\_num\_threads, 695  
 \_\_gnu\_parallel::sequential\_tag, 697  
 \_\_gnu\_parallel::unbalanced\_tag, 697  
 \_\_get\_num\_threads, 698  
 set\_num\_threads, 698  
 \_\_gnu\_pbds, 292  
 \_\_gnu\_pbds::associative\_tag, 698  
 \_\_gnu\_pbds::basic\_branch< Key, Mapped, Tag, Node\_↵  
 Update, Policy\_Tl, \_Alloc >, 699  
 \_\_gnu\_pbds::basic\_branch\_tag, 700  
 \_\_gnu\_pbds::basic\_hash\_table< Key, Mapped, Hash\_↵  
 \_Fn, Eq\_Fn, Resize\_Policy, Store\_Hash, Tag,  
 Policy\_Tl, \_Alloc >, 700

[\\_\\_gnu\\_pbds::basic\\_hash\\_tag](#), [702](#)  
[\\_\\_gnu\\_pbds::basic\\_invalidation\\_guarantee](#), [703](#)  
[\\_\\_gnu\\_pbds::binary\\_heap\\_tag](#), [704](#)  
[\\_\\_gnu\\_pbds::binomial\\_heap\\_tag](#), [705](#)  
[\\_\\_gnu\\_pbds::cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger](#)  
[cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger](#), [706](#)  
[external\\_load\\_access](#), [706](#)  
[get\\_load](#), [707](#)  
[is\\_grow\\_needed](#), [707](#)  
[is\\_resize\\_needed](#), [707](#)  
[notify\\_cleared](#), [707](#)  
[notify\\_erase\\_search\\_collision](#), [707](#)  
[notify\\_erase\\_search\\_end](#), [707](#)  
[notify\\_erase\\_search\\_start](#), [707](#)  
[notify\\_erased](#), [707](#)  
[notify\\_externally\\_resized](#), [708](#)  
[notify\\_find\\_search\\_collision](#), [708](#)  
[notify\\_find\\_search\\_end](#), [708](#)  
[notify\\_find\\_search\\_start](#), [708](#)  
[notify\\_insert\\_search\\_collision](#), [708](#)  
[notify\\_insert\\_search\\_end](#), [708](#)  
[notify\\_insert\\_search\\_start](#), [708](#)  
[notify\\_inserted](#), [709](#)  
[notify\\_resized](#), [709](#)  
[set\\_load](#), [709](#)  
[\\_\\_gnu\\_pbds::cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger](#)  
[trigger](#) < External\_Load\_Access, Size\_Type >, [705](#)  
[\\_\\_gnu\\_pbds::cc\\_hash\\_table](#)  
[cc\\_hash\\_table](#), [711–713](#)  
[\\_\\_gnu\\_pbds::cc\\_hash\\_table](#) < Key, Mapped, Hash\_Fn, Eq\_Fn, Comb\_Hash\_Fn, Resize\_Policy, Store\_Hash, \_Alloc >, [709](#)  
[\\_\\_gnu\\_pbds::cc\\_hash\\_tag](#), [714](#)  
[\\_\\_gnu\\_pbds::container\\_error](#), [715](#)  
[\\_\\_gnu\\_pbds::container\\_tag](#), [715](#)  
[\\_\\_gnu\\_pbds::container\\_traits](#) < Cntnr >, [716](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base](#) < \_Tag >, [717](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base](#) < binary\_heap\_tag >, [717](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base](#) < binomial\_heap\_tag >, [717](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base](#) < cc\_hash\_tag >, [718](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base](#) < gp\_hash\_tag >, [718](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base](#) < list\_update\_tag >, [718](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base](#) < ov\_tree\_tag >, [719](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base](#) < pairing\_heap\_tag >, [719](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base](#) < pat\_trie\_tag >, [720](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base](#) < rb\_tree\_tag >, [720](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base](#) < rc\_binomial\_heap\_tag >, [720](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base](#) < splay\_tree\_tag >, [721](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base](#) < thin\_heap\_tag >, [721](#)  
[\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_const\\_iterator](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc >, [722](#)  
[\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_const\\_node\\_iterator](#)  
[const\\_reference](#), [724](#)  
[difference\\_type](#), [724](#)  
[get\\_l\\_child](#), [725](#)  
[get\\_metadata](#), [725](#)  
[get\\_r\\_child](#), [725](#)  
[iterator\\_category](#), [724](#)  
[metadata\\_const\\_reference](#), [725](#)  
[metadata\\_type](#), [725](#)  
[operator!=](#), [726](#)  
[operator\\*](#), [726](#)  
[operator==](#), [726](#)  
[reference](#), [725](#)  
[value\\_type](#), [725](#)  
[\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_const\\_node\\_iterator](#) < Node, Const\_Iterator, Iterator, \_Alloc >, [723](#)  
[\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_iterator](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc >, [726](#)  
[\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_node\\_iterator](#)  
[const\\_reference](#), [729](#)  
[difference\\_type](#), [729](#)  
[get\\_l\\_child](#), [730](#)  
[get\\_metadata](#), [730](#)  
[get\\_r\\_child](#), [730](#)  
[iterator\\_category](#), [729](#)  
[metadata\\_const\\_reference](#), [729](#)  
[metadata\\_type](#), [729](#)  
[operator!=](#), [730](#)  
[operator\\*](#), [730](#)  
[operator==](#), [731](#)  
[reference](#), [730](#)  
[value\\_type](#), [730](#)  
[\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_node\\_iterator](#) < Node, Const\_Iterator, Iterator, \_Alloc >, [728](#)  
[\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_traits](#)  
[node\\_const\\_iterator](#), [732](#)  
[\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_traits](#) < Key, Mapped, Cmp\_Fn, Node\_Update, Node, \_Alloc >, [731](#)  
[\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_traits](#) < Key, null\_type, Cmp\_Fn, Node\_Update, Node, \_Alloc >, [732](#)  
[node\\_const\\_iterator](#), [732](#)

- `__gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >`, 733
- `__gnu_pbds::detail::binary_heap_const_iterator_`
  - `binary_heap_const_iterator_`, 737
  - `const_pointer`, 736
  - `const_reference`, 736
  - `difference_type`, 736
  - `iterator_category`, 736
  - `operator!=`, 737
  - `operator*`, 737
  - `operator->`, 737
  - `operator==`, 738
  - `pointer`, 736
  - `reference`, 736
  - `value_type`, 737
- `__gnu_pbds::detail::binary_heap_const_iterator_↵`
  - `< Value_Type, Entry, Simple, _Alloc >`, 735
- `__gnu_pbds::detail::binary_heap_point_const_iterator_`
  - `binary_heap_point_const_iterator_`, 740
  - `const_pointer`, 739
  - `const_reference`, 739
  - `difference_type`, 739
  - `iterator_category`, 740
  - `operator!=`, 741
  - `operator*`, 741
  - `operator->`, 741
  - `operator==`, 741
  - `pointer`, 740
  - `reference`, 740
  - `value_type`, 740
- `__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >`, 738
- `__gnu_pbds::detail::binomial_heap< Value_Type, Cmp_↵`
  - `_Fn, _Alloc >`, 741
- `__gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >`, 744
- `__gnu_pbds::detail::branch_policy< Node_Cltr, Node_↵`
  - `Cltr, _Alloc >`, 747
- `__gnu_pbds::detail::branch_policy< Node_Cltr, Node_ltr, _Alloc >`, 746
- `__gnu_pbds::detail::cc_ht_map`
  - `empty`, 751
  - `get_comb_hash_fn`, 751
  - `get_eq_fn`, 751
  - `get_hash_fn`, 752
  - `get_resize_policy`, 752
- `__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_↵`
  - `Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_↵`
  - `_Fn, Resize_Policy >`, 748
- `__gnu_pbds::detail::cond_dealtor< Entry, _Alloc >`, 752
- `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, Tag, Policy_TI >`, 753
- `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_TI >`, 756
  - `type`, 756
- `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_TI >`, 757
  - `type`, 757
- `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_TI >`, 757
  - `type`, 758
- `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_TI >`, 758
  - `type`, 758
- `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_TI >`, 758
- `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_TI >`, 759
  - `type`, 759
- `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_TI >`, 759
  - `type`, 760
- `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_TI >`, 760
  - `type`, 760
- `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_TI >`, 761
  - `type`, 761
- `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_TI >`, 761
  - `type`, 761
- `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_TI >`, 762
  - `type`, 762
- `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_TI >`, 762
  - `type`, 763
- `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_TI >`, 763
  - `type`, 763
- `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_TI >`, 763
  - `type`, 763
- `__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type >`, 753
  - `type`, 754
- `__gnu_pbds::detail::container_base_dispatch< _VTp,`

Cmp\_Fn, \_Alloc, binomial\_heap\_tag, null\_type  
 >, [754](#)  
 type, [754](#)  
 \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp,  
 Cmp\_Fn, \_Alloc, pairing\_heap\_tag, null\_type  
 >, [754](#)  
 type, [755](#)  
 \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp,  
 Cmp\_Fn, \_Alloc, rc\_binomial\_heap\_tag, null\_↵  
 type >, [755](#)  
 type, [755](#)  
 \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp,  
 Cmp\_Fn, \_Alloc, thin\_heap\_tag, null\_type >,  
[756](#)  
 type, [756](#)  
 \_\_gnu\_pbds::detail::default\_comb\_hash\_fn, [764](#)  
 type, [764](#)  
 \_\_gnu\_pbds::detail::default\_eq\_fn  
 type, [765](#)  
 \_\_gnu\_pbds::detail::default\_eq\_fn< Key >, [764](#)  
 \_\_gnu\_pbds::detail::default\_hash\_fn  
 type, [765](#)  
 \_\_gnu\_pbds::detail::default\_hash\_fn< Key >, [765](#)  
 \_\_gnu\_pbds::detail::default\_probe\_fn  
 type, [766](#)  
 \_\_gnu\_pbds::detail::default\_probe\_fn< Comb\_Probe\_Fn  
 >, [765](#)  
 \_\_gnu\_pbds::detail::default\_resize\_policy  
 type, [766](#)  
 \_\_gnu\_pbds::detail::default\_resize\_policy< Comb\_↵  
 Hash\_Fn >, [766](#)  
 \_\_gnu\_pbds::detail::default\_trie\_access\_traits< Key >,  
[766](#)  
 \_\_gnu\_pbds::detail::default\_trie\_access\_traits< std\_↵  
 ::basic\_string< Char, Char\_Traits, std\_↵  
 ::allocator< char > > >, [767](#)  
 type, [767](#)  
 \_\_gnu\_pbds::detail::default\_update\_policy, [767](#)  
 type, [767](#)  
 \_\_gnu\_pbds::detail::dumnode\_const\_iterator< Key, Data,  
 \_Alloc >, [768](#)  
 \_\_gnu\_pbds::detail::entry\_cmp< \_VTp, Cmp\_Fn, \_Alloc,  
 No\_Throw >, [768](#)  
 \_\_gnu\_pbds::detail::entry\_cmp< \_VTp, Cmp\_Fn, \_Alloc,  
 false >, [768](#)  
 \_\_gnu\_pbds::detail::entry\_cmp< \_VTp, Cmp\_Fn, \_Alloc,  
 false >::type, [769](#)  
 \_\_gnu\_pbds::detail::entry\_cmp< \_VTp, Cmp\_Fn, \_Alloc,  
 true >, [769](#)  
 type, [769](#)  
 \_\_gnu\_pbds::detail::entry\_pred< \_VTp, Pred, \_Alloc,  
 No\_Throw >, [770](#)  
 \_\_gnu\_pbds::detail::entry\_pred< \_VTp, Pred, \_Alloc, false  
 >, [770](#)  
 \_\_gnu\_pbds::detail::entry\_pred< \_VTp, Pred, \_Alloc, true  
 >, [770](#)  
 \_\_gnu\_pbds::detail::eq\_by\_less< Key, Cmp\_Fn >, [771](#)  
 \_\_gnu\_pbds::detail::gp\_ht\_map  
 empty, [774](#)  
 get\_comb\_probe\_fn, [774](#)  
 get\_eq\_fn, [774](#)  
 get\_hash\_fn, [774](#)  
 get\_probe\_fn, [775](#)  
 get\_resize\_policy, [775](#)  
 \_\_gnu\_pbds::detail::gp\_ht\_map< Key, Mapped, Hash\_↵  
 Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_↵  
 Fn, Probe\_Fn, Resize\_Policy >, [771](#)  
 \_\_gnu\_pbds::detail::hash\_eq\_fn< Key, Eq\_Fn, \_Alloc,  
 Store\_Hash >, [776](#)  
 \_\_gnu\_pbds::detail::hash\_eq\_fn< Key, Eq\_Fn, \_Alloc,  
 false >, [776](#)  
 \_\_gnu\_pbds::detail::hash\_eq\_fn< Key, Eq\_Fn, \_Alloc,  
 true >, [777](#)  
 \_\_gnu\_pbds::detail::hash\_load\_check\_resize\_trigger\_↵  
 size\_base< Size\_Type, Hold\_Size >, [777](#)  
 \_\_gnu\_pbds::detail::hash\_load\_check\_resize\_trigger\_↵  
 size\_base< Size\_Type, true >, [778](#)  
 \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap< Value\_↵  
 \_Type, Cmp\_Fn, Node\_Metadata, \_Alloc >, [778](#)  
 \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_↵  
 iterator\_  
 const\_pointer, [781](#)  
 const\_reference, [781](#)  
 difference\_type, [781](#)  
 iterator\_category, [781](#)  
 left\_child\_next\_sibling\_heap\_const\_iterator\_, [782](#)  
 operator!=, [782](#)  
 operator\*, [782](#)  
 operator->, [782](#)  
 operator==, [783](#)  
 pointer, [781](#)  
 reference, [781](#)  
 value\_type, [781](#)  
 \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_↵  
 iterator\_< Node, \_Alloc >, [780](#)  
 \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_↵  
 \_< \_Value, \_Metadata, \_Alloc >, [783](#)  
 \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_↵  
 \_point\_const\_iterator\_  
 const\_pointer, [785](#)  
 const\_reference, [785](#)  
 difference\_type, [785](#)  
 iterator\_category, [785](#)  
 left\_child\_next\_sibling\_heap\_node\_point\_const\_↵  
 iterator\_, [786](#)  
 operator!=, [786](#)  
 operator\*, [786](#)  
 operator->, [786](#)

- operator==, [787](#)
- pointer, [785](#)
- reference, [785](#)
- value\_type, [786](#)
- \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node↵
  - \_point\_const\_iterator< Node, \_Alloc >, [784](#)
- \_\_gnu\_pbds::detail::lu\_counter\_metadata< Size\_Type >, [787](#)
- \_\_gnu\_pbds::detail::lu\_counter\_policy\_base< Size\_Type >, [787](#)
- \_\_gnu\_pbds::detail::lu\_map< Key, Mapped, Eq\_Fn, \_↵
  - Alloc, Update\_Policy >, [788](#)
- \_\_gnu\_pbds::detail::mask\_based\_range\_hashing< Size\_Type >, [790](#)
- \_\_gnu\_pbds::detail::mod\_based\_range\_hashing< Size\_↵
  - \_Type >, [791](#)
- \_\_gnu\_pbds::detail::no\_throw\_copies< Key, Mapped >, [792](#)
- \_\_gnu\_pbds::detail::no\_throw\_copies< Key, null\_type >, [792](#)
- \_\_gnu\_pbds::detail::ov\_tree\_map
  - node\_begin, [795](#)
  - node\_end, [795](#)
- \_\_gnu\_pbds::detail::ov\_tree\_map< Key, Mapped, Cmp\_↵
  - \_Fn, Node\_And\_It\_Traits, \_Alloc >, [793](#)
- \_\_gnu\_pbds::detail::ov\_tree\_map< Key, Mapped, Cmp\_↵
  - \_Fn, Node\_And\_It\_Traits, \_Alloc >::cond\_dtor< Size\_Type >, [796](#)
- \_\_gnu\_pbds::detail::ov\_tree\_node\_const\_it\_
  - get\_l\_child, [798](#)
  - get\_r\_child, [798](#)
- \_\_gnu\_pbds::detail::ov\_tree\_node\_const\_it\_< Value\_↵
  - Type, Metadata\_Type, \_Alloc >, [796](#)
- \_\_gnu\_pbds::detail::ov\_tree\_node\_it\_
  - get\_l\_child, [799](#)
  - get\_r\_child, [799](#)
  - operator\*, [799](#)
- \_\_gnu\_pbds::detail::ov\_tree\_node\_it\_< Value\_Type, Metadata\_Type, \_Alloc >, [798](#)
- \_\_gnu\_pbds::detail::pairing\_heap< Value\_Type, Cmp\_Fn, \_Alloc >, [800](#)
- \_\_gnu\_pbds::detail::pat\_trie\_base, [802](#)
  - node\_type, [803](#)
- \_\_gnu\_pbds::detail::pat\_trie\_base::\_CIter< Node, Leaf, Head, Inode, Is\_Forward\_Iterator >, [803](#)
- \_\_gnu\_pbds::detail::pat\_trie\_base::\_Head< \_ATraits, Metadata >, [805](#)
- \_\_gnu\_pbds::detail::pat\_trie\_base::\_Inode< \_ATraits, Metadata >, [806](#)
- \_\_gnu\_pbds::detail::pat\_trie\_base::\_Inode< \_ATraits, Metadata >::const\_iterator, [808](#)
- \_\_gnu\_pbds::detail::pat\_trie\_base::\_Inode< \_ATraits, Metadata >::iterator, [809](#)
- \_\_gnu\_pbds::detail::pat\_trie\_base::\_Iter< Node, Leaf, Head, Inode, Is\_Forward\_Iterator >, [810](#)
- \_\_gnu\_pbds::detail::pat\_trie\_base::\_Leaf< \_ATraits, Metadata >, [812](#)
- \_\_gnu\_pbds::detail::pat\_trie\_base::\_Metadata< Meta- data, \_Alloc >, [813](#)
- \_\_gnu\_pbds::detail::pat\_trie\_base::\_Metadata< null\_↵
  - type, \_Alloc >, [814](#)
- \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_base< \_A\_↵
  - Traits, Metadata >, [814](#)
- \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer
  - \_\_rebind\_m, [817](#)
  - get\_child, [817](#)
  - get\_metadata, [817](#)
  - metadata\_type, [817](#)
  - num\_children, [817](#)
  - operator!=, [817](#)
  - operator\*, [818](#)
  - operator==, [818](#)
  - valid\_prefix, [818](#)
- \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer< Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc >, [815](#)
- \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_iter
  - \_\_rebind\_m, [820](#)
  - get\_child, [820](#)
  - get\_metadata, [820](#)
  - metadata\_type, [820](#)
  - num\_children, [820](#)
  - operator!=, [820](#)
  - operator\*, [821](#)
  - operator==, [821](#)
  - valid\_prefix, [821](#)
- \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_iter< Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc >, [818](#)
- \_\_gnu\_pbds::detail::pat\_trie\_map
  - node\_begin, [824](#)
  - node\_end, [824](#)
  - node\_type, [824](#)
- \_\_gnu\_pbds::detail::pat\_trie\_map< Key, Mapped, Node\_↵
  - \_And\_It\_Traits, \_Alloc >, [821](#)
- \_\_gnu\_pbds::detail::probe\_fn\_base< \_Alloc >, [824](#)
- \_\_gnu\_pbds::detail::ranged\_hash\_fn< Key, Hash\_Fn, \_↵
  - Alloc, Comb\_Hash\_Fn, Store\_Hash >, [825](#)
- \_\_gnu\_pbds::detail::ranged\_hash\_fn< Key, Hash\_Fn, \_↵
  - Alloc, Comb\_Hash\_Fn, false >, [825](#)
- \_\_gnu\_pbds::detail::ranged\_hash\_fn< Key, Hash\_Fn, \_↵
  - Alloc, Comb\_Hash\_Fn, true >, [826](#)
- \_\_gnu\_pbds::detail::ranged\_hash\_fn< Key, null\_type, \_↵
  - Alloc, Comb\_Hash\_Fn, false >, [827](#)
- \_\_gnu\_pbds::detail::ranged\_hash\_fn< Key, null\_type, \_↵
  - Alloc, Comb\_Hash\_Fn, true >, [827](#)
- \_\_gnu\_pbds::detail::ranged\_probe\_fn< Key, Hash\_Fn, \_↵
  - \_Alloc, Comb\_Probe\_Fn, Probe\_Fn, Store\_↵

- Hash >, [828](#)
- \_\_gnu\_pbds::detail::ranged\_probe\_fn< Key, Hash\_Fn, ↵  
\_Alloc, Comb\_Probe\_Fn, Probe\_Fn, false >, [829](#)
- \_\_gnu\_pbds::detail::ranged\_probe\_fn< Key, Hash\_Fn, ↵  
\_Alloc, Comb\_Probe\_Fn, Probe\_Fn, true >, [829](#)
- \_\_gnu\_pbds::detail::ranged\_probe\_fn< Key, null\_type, ↵  
\_Alloc, Comb\_Probe\_Fn, null\_type, false >, [830](#)
- \_\_gnu\_pbds::detail::rb\_tree\_map  
node\_begin, [834](#)  
node\_end, [834](#)
- \_\_gnu\_pbds::detail::rb\_tree\_map< Key, Mapped, Cmp\_↵  
Fn, Node\_And\_It\_Traits, \_Alloc >, [831](#)
- \_\_gnu\_pbds::detail::rb\_tree\_node< Value\_Type, Meta-  
data, \_Alloc >, [834](#)
- \_\_gnu\_pbds::detail::rc< \_Node, \_Alloc >, [835](#)
- \_\_gnu\_pbds::detail::rc\_binomial\_heap< Value\_Type,  
Cmp\_Fn, \_Alloc >, [836](#)
- \_\_gnu\_pbds::detail::resize\_policy< \_Tp >, [838](#)
- \_\_gnu\_pbds::detail::splay\_tree\_map  
node\_begin, [842](#)  
node\_end, [842](#)
- \_\_gnu\_pbds::detail::splay\_tree\_map< Key, Mapped,  
Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >, [839](#)
- \_\_gnu\_pbds::detail::splay\_tree\_node< Value\_Type,  
Metadata, \_Alloc >, [842](#)
- \_\_gnu\_pbds::detail::stored\_data< \_Tv, \_Th >, [844](#)
- \_\_gnu\_pbds::detail::stored\_data< \_Tv, null\_type >, [845](#)
- \_\_gnu\_pbds::detail::stored\_hash< \_Th >, [846](#)
- \_\_gnu\_pbds::detail::stored\_value< \_Tv >, [847](#)
- \_\_gnu\_pbds::detail::synth\_access\_traits< Type\_Traits,  
Set, \_ATraits >, [847](#)
- \_\_gnu\_pbds::detail::thin\_heap< Value\_Type, Cmp\_Fn, ↵  
\_Alloc >, [848](#)
- \_\_gnu\_pbds::detail::tree\_metadata\_helper< Node\_↵  
Update, \_BTp >, [850](#)
- \_\_gnu\_pbds::detail::tree\_metadata\_helper< Node\_↵  
Update, false >, [850](#)
- \_\_gnu\_pbds::detail::tree\_metadata\_helper< Node\_↵  
Update, true >, [851](#)
- \_\_gnu\_pbds::detail::tree\_node\_metadata\_dispatch< Key,  
Data, Cmp\_Fn, Node\_Update, \_Alloc >, [851](#)
- \_\_gnu\_pbds::detail::tree\_traits< Key, Data, Cmp\_Fn,  
Node\_Update, Tag, \_Alloc >, [852](#)
- \_\_gnu\_pbds::detail::tree\_traits< Key, Mapped, Cmp\_Fn,  
Node\_Update, ov\_tree\_tag, \_Alloc >, [852](#)  
node\_const\_iterator, [852](#)
- \_\_gnu\_pbds::detail::tree\_traits< Key, Mapped, Cmp\_Fn,  
Node\_Update, rb\_tree\_tag, \_Alloc >, [853](#)  
node\_const\_iterator, [854](#)
- \_\_gnu\_pbds::detail::tree\_traits< Key, Mapped, Cmp\_Fn,  
Node\_Update, splay\_tree\_tag, \_Alloc >, [854](#)  
node\_const\_iterator, [855](#)
- \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn,  
Node\_Update, ov\_tree\_tag, \_Alloc >, [855](#)  
node\_const\_iterator, [856](#)
- \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn,  
Node\_Update, rb\_tree\_tag, \_Alloc >, [856](#)  
node\_const\_iterator, [857](#)
- \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn,  
Node\_Update, splay\_tree\_tag, \_Alloc >, [858](#)  
node\_const\_iterator, [859](#)
- \_\_gnu\_pbds::detail::trie\_metadata\_helper< Node\_↵  
Update, \_BTp >, [859](#)
- \_\_gnu\_pbds::detail::trie\_metadata\_helper< Node\_↵  
Update, false >, [859](#)
- \_\_gnu\_pbds::detail::trie\_metadata\_helper< Node\_↵  
Update, true >, [860](#)
- \_\_gnu\_pbds::detail::trie\_node\_metadata\_dispatch< Key,  
Data, Cmp\_Fn, Node\_Update, \_Alloc >, [860](#)
- \_\_gnu\_pbds::detail::trie\_policy\_base< Node\_Cltr,  
Node\_Itr, \_ATraits, \_Alloc >, [861](#)
- \_\_gnu\_pbds::detail::trie\_traits< Key, Data, \_ATraits,  
Node\_Update, Tag, \_Alloc >, [862](#)
- \_\_gnu\_pbds::detail::trie\_traits< Key, Mapped, \_ATraits,  
Node\_Update, pat\_trie\_tag, \_Alloc >, [862](#)  
node\_const\_iterator, [863](#)  
node\_update, [863](#)  
synth\_access\_traits, [863](#)
- \_\_gnu\_pbds::detail::trie\_traits< Key, null\_type, \_ATraits,  
Node\_Update, pat\_trie\_tag, \_Alloc >, [864](#)  
node\_const\_iterator, [864](#)  
node\_update, [864](#)  
synth\_access\_traits, [864](#)
- \_\_gnu\_pbds::detail::type\_base< Key, Mapped, \_Alloc,  
Store\_Hash >, [865](#)
- \_\_gnu\_pbds::detail::type\_base< Key, Mapped, \_Alloc,  
false >, [866](#)
- \_\_gnu\_pbds::detail::type\_base< Key, Mapped, \_Alloc,  
true >, [866](#)
- \_\_gnu\_pbds::detail::type\_base< Key, null\_type, \_Alloc,  
false >, [867](#)
- \_\_gnu\_pbds::detail::type\_base< Key, null\_type, \_Alloc,  
true >, [867](#)
- \_\_gnu\_pbds::detail::type\_dispatch< Key, Mapped, \_Alloc,  
Store\_Hash >, [868](#)
- \_\_gnu\_pbds::detail::types\_traits< Key, Mapped, \_Alloc,  
Store\_Hash >, [869](#)
- \_\_gnu\_pbds::direct\_mask\_range\_hashing  
operator(), [870](#)
- \_\_gnu\_pbds::direct\_mask\_range\_hashing< Size\_Type >,  
[870](#)
- \_\_gnu\_pbds::direct\_mod\_range\_hashing  
operator(), [872](#)
- \_\_gnu\_pbds::direct\_mod\_range\_hashing< Size\_Type >,  
[871](#)
- \_\_gnu\_pbds::gp\_hash\_table  
gp\_hash\_table, [873–876](#)



- \_\_gnu\_pbds::gp\_hash\_table< Key, Mapped, Hash\_Fn, Eq\_Fn, Comb\_Probe\_Fn, Probe\_Fn, Resize\_↵ Policy, Store\_Hash, \_Alloc >, 872
- \_\_gnu\_pbds::gp\_hash\_tag, 877
- \_\_gnu\_pbds::hash\_exponential\_size\_policy hash\_exponential\_size\_policy, 878
- \_\_gnu\_pbds::hash\_exponential\_size\_policy< Size\_Type >, 877
- \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger
  - external\_load\_access, 879
  - get\_loads, 880
  - hash\_load\_check\_resize\_trigger, 879
  - notify\_cleared, 880
  - notify\_inserted, 880
  - notify\_resized, 880
  - set\_loads, 880
- \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger< External↵ \_Load\_Access, Size\_Type >, 878
- \_\_gnu\_pbds::hash\_prime\_size\_policy, 880
  - hash\_prime\_size\_policy, 881
  - size\_type, 881
- \_\_gnu\_pbds::hash\_standard\_resize\_policy
  - get\_actual\_size, 883
  - get\_new\_size, 883
  - get\_size\_policy, 883
  - get\_trigger\_policy, 883
  - hash\_standard\_resize\_policy, 882
  - resize, 884
- \_\_gnu\_pbds::hash\_standard\_resize\_policy< Size\_Policy, Trigger\_Policy, External\_Size\_Access, Size\_↵ Type >, 881
- \_\_gnu\_pbds::insert\_error, 884
- \_\_gnu\_pbds::join\_error, 885
- \_\_gnu\_pbds::linear\_probe\_fn
  - operator(), 886
- \_\_gnu\_pbds::linear\_probe\_fn< Size\_Type >, 885
- \_\_gnu\_pbds::list\_update
  - list\_update, 886
- \_\_gnu\_pbds::list\_update< Key, Mapped, Eq\_Fn, Update\_Policy, \_Alloc >, 886
- \_\_gnu\_pbds::list\_update\_tag, 887
- \_\_gnu\_pbds::lu\_counter\_policy
  - max\_count, 889
  - metadata\_reference, 889
  - metadata\_type, 889
  - operator(), 889
- \_\_gnu\_pbds::lu\_counter\_policy< Max\_Count, \_Alloc >, 888
- \_\_gnu\_pbds::lu\_move\_to\_front\_policy
  - metadata\_reference, 890
  - metadata\_type, 890
  - operator(), 890
- \_\_gnu\_pbds::lu\_move\_to\_front\_policy< \_Alloc >, 889
- \_\_gnu\_pbds::null\_node\_update< \_Tp1, \_Tp2, \_Tp3, \_↵ Tp4 >, 891
- \_\_gnu\_pbds::null\_type, 891
- \_\_gnu\_pbds::ov\_tree\_tag, 892
- \_\_gnu\_pbds::pairing\_heap\_tag, 893
- \_\_gnu\_pbds::pat\_trie\_tag, 894
- \_\_gnu\_pbds::point\_invalidation\_guarantee, 895
- \_\_gnu\_pbds::priority\_queue< \_Tv, Cmp\_Fn, Tag, \_Alloc >, 895
- \_\_gnu\_pbds::priority\_queue\_tag, 897
- \_\_gnu\_pbds::quadratic\_probe\_fn
  - operator(), 898
- \_\_gnu\_pbds::quadratic\_probe\_fn< Size\_Type >, 897
- \_\_gnu\_pbds::range\_invalidation\_guarantee, 898
- \_\_gnu\_pbds::rb\_tree\_tag, 899
- \_\_gnu\_pbds::rc\_binomial\_heap\_tag, 900
- \_\_gnu\_pbds::resize\_error, 901
- \_\_gnu\_pbds::sample\_probe\_fn, 901
  - operator(), 902
  - sample\_probe\_fn, 902
  - swap, 902
- \_\_gnu\_pbds::sample\_range\_hashing, 902
  - notify\_resized, 903
  - operator(), 903
  - sample\_range\_hashing, 903
  - size\_type, 903
  - swap, 903
- \_\_gnu\_pbds::sample\_ranged\_hash\_fn, 903
  - notify\_resized, 904
  - operator(), 904
  - sample\_ranged\_hash\_fn, 904
  - swap, 904
- \_\_gnu\_pbds::sample\_ranged\_probe\_fn, 904
- \_\_gnu\_pbds::sample\_resize\_policy, 905
  - get\_new\_size, 906
  - is\_resize\_needed, 906
  - notify\_cleared, 906
  - notify\_erase\_search\_collision, 906
  - notify\_erase\_search\_end, 906
  - notify\_erase\_search\_start, 906
  - notify\_erased, 906
  - notify\_find\_search\_collision, 906
  - notify\_find\_search\_end, 906
  - notify\_find\_search\_start, 906
  - notify\_insert\_search\_collision, 906
  - notify\_insert\_search\_end, 906
  - notify\_insert\_search\_start, 906
  - notify\_inserted, 907
  - notify\_resized, 907
  - sample\_range\_hashing, 907
  - sample\_resize\_policy, 906
  - size\_type, 905
  - swap, 907
- \_\_gnu\_pbds::sample\_resize\_trigger, 907

- is\_grow\_needed, 908
- is\_resize\_needed, 908
- notify\_cleared, 908
- notify\_erase\_search\_collision, 908
- notify\_erase\_search\_end, 908
- notify\_erase\_search\_start, 908
- notify\_erased, 908
- notify\_externally\_resized, 908
- notify\_find\_search\_collision, 908
- notify\_find\_search\_end, 909
- notify\_find\_search\_start, 909
- notify\_insert\_search\_collision, 909
- notify\_insert\_search\_end, 909
- notify\_insert\_search\_start, 909
- notify\_inserted, 909
- notify\_resized, 909
- sample\_range\_hashing, 909
- sample\_resize\_trigger, 908
- size\_type, 908
- swap, 909
- \_\_gnu\_pbds::sample\_size\_policy, 909
  - get\_nearest\_larger\_size, 910
  - get\_nearest\_smaller\_size, 910
  - sample\_range\_hashing, 910
  - sample\_size\_policy, 910
  - size\_type, 910
  - swap, 910
- \_\_gnu\_pbds::sample\_tree\_node\_update< Const\_Node\_↵  
\_Iter, Node\_Itr, Cmp\_Fn, \_Alloc >, 910
- \_\_gnu\_pbds::sample\_trie\_access\_traits, 911
  - begin, 911
  - e\_pos, 911
  - e\_type, 911
  - end, 912
- \_\_gnu\_pbds::sample\_trie\_node\_update
  - operator(), 912
  - sample\_trie\_node\_update, 912
- \_\_gnu\_pbds::sample\_trie\_node\_update< Node\_Cltr, Node\_Itr, ATraits, \_Alloc >, 912
- \_\_gnu\_pbds::sample\_update\_policy, 913
  - metadata\_type, 913
  - operator(), 913
  - sample\_update\_policy, 913
  - swap, 913
- \_\_gnu\_pbds::sequence\_tag, 914
- \_\_gnu\_pbds::splay\_tree\_tag, 915
- \_\_gnu\_pbds::string\_tag, 916
- \_\_gnu\_pbds::thin\_heap\_tag, 917
- \_\_gnu\_pbds::tree
  - cmp\_fn, 918
  - tree, 918, 919
- \_\_gnu\_pbds::tree< Key, Mapped, Cmp\_Fn, Tag, Node\_↵  
Update, \_Alloc >, 917
- \_\_gnu\_pbds::tree\_order\_statistics\_node\_update
  - find\_by\_order, 921
  - operator(), 921
  - order\_of\_key, 921
- \_\_gnu\_pbds::tree\_order\_statistics\_node\_update< Node\_↵  
\_Cltr, Node\_Itr, Cmp\_Fn, \_Alloc >, 919
- \_\_gnu\_pbds::tree\_tag, 922
- \_\_gnu\_pbds::trie
  - access\_traits, 923
  - trie, 923, 924
- \_\_gnu\_pbds::trie< Key, Mapped, ATraits, Tag, Node\_↵  
Update, \_Alloc >, 922
- \_\_gnu\_pbds::trie\_order\_statistics\_node\_update
  - find\_by\_order, 926
  - operator(), 927
  - order\_of\_key, 927
  - order\_of\_prefix, 927
- \_\_gnu\_pbds::trie\_order\_statistics\_node\_update< Node\_↵  
\_Cltr, Node\_Itr, ATraits, \_Alloc >, 925
- \_\_gnu\_pbds::trie\_prefix\_search\_node\_update
  - a\_const\_iterator, 929
  - access\_traits, 929
  - allocator\_type, 929
  - operator(), 930
  - prefix\_range, 930
  - size\_type, 930
- \_\_gnu\_pbds::trie\_prefix\_search\_node\_update< Node\_↵  
\_Cltr, Node\_Itr, ATraits, \_Alloc >, 928
- \_\_gnu\_pbds::trie\_string\_access\_traits
  - begin, 932
  - const\_iterator, 932
  - e\_pos, 932
  - e\_type, 932
  - end, 932
- \_\_gnu\_pbds::trie\_string\_access\_traits< String, Min\_E\_↵  
Val, Max\_E\_Val, Reverse, \_Alloc >, 931
- \_\_gnu\_pbds::trie\_tag, 933
- \_\_gnu\_pbds::trivial\_iterator\_tag, 933
- \_\_gnu\_profile, 295
  - \_GLIBCXX\_PROFILE\_DEFINE\_UNINIT\_DATA, 299
  - \_env\_t, 298
  - \_profcxx\_init, 298
  - \_report, 298
- \_\_gnu\_profile::\_\_container\_size\_info, 934
- \_\_gnu\_profile::\_\_container\_size\_stack\_info, 935
- \_\_gnu\_profile::\_\_hashfunc\_info, 936
- \_\_gnu\_profile::\_\_hashfunc\_stack\_info, 937
- \_\_gnu\_profile::\_\_list2vector\_info, 938
- \_\_gnu\_profile::\_\_map2umap\_info, 939
- \_\_gnu\_profile::\_\_map2umap\_stack\_info, 940
- \_\_gnu\_profile::\_\_object\_info\_base, 941
- \_\_gnu\_profile::\_\_reentrance\_guard, 942
- \_\_gnu\_profile::\_\_stack\_hash, 942
- \_\_gnu\_profile::\_\_stack\_info\_base< \_\_object\_info >, 942



- `__gnu_profile::__trace_base< __object_info, __stack_info >`, 943
- `__gnu_profile::__trace_container_size`, 943
- `__gnu_profile::__trace_hash_func`, 944
- `__gnu_profile::__trace_hashtable_size`, 945
- `__gnu_profile::__trace_map2umap`, 946
- `__gnu_profile::__trace_vector_size`, 947
- `__gnu_profile::__trace_vector_to_list`, 948
- `__gnu_profile::__vector2list_info`, 949
- `__gnu_profile::__vector2list_stack_info`, 950
- `__gnu_profile::__warning_data`, 951
- `__gnu_sequential`, 299
- `__heap_select`
  - `std`, 356
- `__init_winner`
  - `__gnu_parallel::__LoserTree< false, _Tp, _Compare >`, 639
- `__inner_product_selector`
  - `__gnu_parallel::__inner_product_selector`, 605
- `__inplace_stable_partition`
  - `std`, 356
- `__inplace_stable_sort`
  - `std`, 356
- `__insert_start`
  - `__gnu_parallel::__LoserTree`, 637
  - `__gnu_parallel::__LoserTree< false, _Tp, _Compare >`, 639
  - `__gnu_parallel::__LoserTreeBase`, 643
- `__insertion_sort`
  - `std`, 356, 357
- `__introsort_loop`
  - `std`, 357
- `__is_sorted`
  - `__gnu_parallel`, 262
- `__iterator_category`
  - Iterators, 65
- `__lg`
  - `std`, 357
- `__match_flag`
  - Regular Expressions, 146
- `__median_of_three_iterators`
  - `__gnu_parallel`, 263
- `__merge_adaptive`
  - `std`, 357
- `__merge_advance`
  - `__gnu_parallel`, 263
- `__merge_advance_movc`
  - `__gnu_parallel`, 263
- `__merge_advance_usual`
  - `__gnu_parallel`, 264
- `__merge_without_buffer`
  - `std`, 358
- `__move_median_to_first`
  - `std`, 358
- `__move_merge`
  - `std`, 358
- `__move_merge_adaptive`
  - `std`, 359
- `__move_merge_adaptive_backward`
  - `std`, 359
- `__new_val`
  - `__gnu_parallel::__replace_if_selector`, 614
  - `__gnu_parallel::__replace_selector`, 615
- `__num_bitmaps`
  - `__gnu_cxx::__detail`, 239
- `__num_blocks`
  - `__gnu_cxx::__detail`, 239
- `__num_get_type`
  - `std::basic_ios`, 1106
- `__num_put_type`
  - `std::basic_ios`, 1106
- `__parallel_merge_advance`
  - `__gnu_parallel`, 264, 265
- `__parallel_nth_element`
  - `__gnu_parallel`, 265
- `__parallel_partial_sort`
  - `__gnu_parallel`, 266
- `__parallel_partial_sum`
  - `__gnu_parallel`, 266
- `__parallel_partial_sum_basecase`
  - `__gnu_parallel`, 266
- `__parallel_partial_sum_linear`
  - `__gnu_parallel`, 267
- `__parallel_partition`
  - `__gnu_parallel`, 267
- `__parallel_random_shuffle`
  - `__gnu_parallel`, 268
- `__parallel_random_shuffle_drs`
  - `__gnu_parallel`, 268
- `__parallel_random_shuffle_drs_pu`
  - `__gnu_parallel`, 268
- `__parallel_sort`
  - `__gnu_parallel`, 269, 270, 272, 274
- `__parallel_sort_qs`
  - `__gnu_parallel`, 275
- `__parallel_sort_qs_conquer`
  - `__gnu_parallel`, 275
- `__parallel_sort_qs_divide`
  - `__gnu_parallel`, 275
- `__parallel_sort_qsb`
  - `__gnu_parallel`, 276
- `__parallel_unique_copy`
  - `__gnu_parallel`, 276
- `__partition`
  - `std`, 359
- `__profcxx_init`
  - `__gnu_profile`, 298
- `__qsb_conquer`

- \_\_gnu\_parallel, 277
- \_\_qsb\_divide
  - \_\_gnu\_parallel, 277
- \_\_qsb\_local\_sort\_with\_helping
  - \_\_gnu\_parallel, 277
- \_\_random\_number\_pow2
  - \_\_gnu\_parallel, 279
- \_\_rd\_log2
  - \_\_gnu\_parallel, 279
- \_\_rebind\_m
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer, 817
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_iter, 820
- \_\_replace\_if\_selector
  - \_\_gnu\_parallel::\_replace\_if\_selector, 612
- \_\_replace\_selector
  - \_\_gnu\_parallel::\_replace\_selector, 615
- \_\_report
  - \_\_gnu\_profile, 298
- \_\_reverse
  - std, 360
- \_\_rotate
  - std, 360
- \_\_rotate\_adaptive
  - std, 360
- \_\_round\_up\_to\_pow2
  - \_\_gnu\_parallel, 279
- \_\_search\_n
  - std, 360, 361
- \_\_search\_template
  - \_\_gnu\_parallel, 279
- \_\_sequential\_multiway\_merge
  - \_\_gnu\_parallel, 281
- \_\_sequential\_random\_shuffle
  - \_\_gnu\_parallel, 281
- \_\_shrink
  - \_\_gnu\_parallel, 281
- \_\_shrink\_and\_double
  - \_\_gnu\_parallel, 282
- \_\_stable\_partition\_adaptive
  - std, 361
- \_\_static\_pointer\_cast
  - \_\_gnu\_cxx, 229
- \_\_syntax\_option
  - Regular Expressions, 146
- \_\_umap\_traits
  - std, 353
- \_\_ummap\_traits
  - std, 353
- \_\_umset\_traits
  - std, 353
- \_\_unguarded\_insertion\_sort
  - std, 361
- \_\_unguarded\_linear\_insert
  - std, 362
- \_\_unguarded\_partition
  - std, 362
- \_\_unguarded\_partition\_pivot
  - std, 362
- \_\_unique\_copy
  - std, 363
- \_\_uset\_traits
  - std, 353
- \_\_valid\_range
  - \_\_gnu\_debug, 244, 245
- \_\_valid\_range\_aux
  - \_\_gnu\_debug, 245
- \_\_valid\_range\_aux2
  - \_\_gnu\_debug, 245
- \_\_versa\_string
  - \_\_gnu\_cxx::\_versa\_string, 455–458
- \_\_yield
  - \_\_gnu\_parallel, 282
- ~\_LoserTreeBase
  - \_\_gnu\_parallel::\_LoserTreeBase, 643
- ~\_RestrictedBoundedConcurrentQueue
  - \_\_gnu\_parallel::\_RestrictedBoundedConcurrentQueue, 666
- ~\_Safe\_sequence\_base
  - \_\_gnu\_debug::\_Safe\_sequence\_base, 575
- ~\_Safe\_unordered\_container\_base
  - \_\_gnu\_debug::\_Safe\_unordered\_container\_base, 582
- ~\_\_versa\_string
  - \_\_gnu\_cxx::\_versa\_string, 458
- ~auto\_ptr
  - std::auto\_ptr, 1097
- ~basic\_ios
  - std::basic\_ios, 1109
- ~basic\_regex
  - std::basic\_regex, 1127
- ~basic\_string
  - std::basic\_string, 1138
- ~collate
  - std::collate, 1225
- ~ctype
  - std::ctype< char >, 1251
  - std::ctype< wchar\_t >, 1263
- ~deque
  - std::deque, 1309
- ~facet
  - std::locale::facet, 1459
- ~forward\_list
  - std::forward\_list, 1351
- ~gslice
  - Numeric\_arrays, 115
- ~ios\_base
  - std::ios\_base, 1410
- ~locale

- std::locale, 1454
- ~match\_results
  - std::match\_results, 1491
- ~messages
  - std::messages, 1507
- ~money\_get
  - std::money\_get, 1516
- ~money\_put
  - std::money\_put, 1520
- ~moneypunct
  - std::moneypunct, 1526
- ~num\_get
  - std::num\_get, 1591
- ~num\_put
  - std::num\_put, 1605
- ~numpunct
  - std::numpunct, 1618
- ~stdio\_filebuf
  - \_\_gnu\_cxx::stdio\_filebuf, 535
- ~time\_get
  - std::time\_get, 1728
- ~time\_put
  - std::time\_put, 1743
- ~vector
  - std::vector, 1846
- a
  - std::extreme\_value\_distribution, 1339
  - std::weibull\_distribution, 1864
- a\_const\_iterator
  - \_\_gnu\_pbds::trie\_prefix\_search\_node\_update, 929
- ATOMIC\_BOOL\_LOCK\_FREE
  - Atomics, 13
- abi, 299
- access\_traits
  - \_\_gnu\_pbds::trie, 923
  - \_\_gnu\_pbds::trie\_prefix\_search\_node\_update, 929
- accumulate
  - std, 364
- accumulate\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 669
- Adaptors for pointers to functions, 4
  - ptr\_fun, 5
- Adaptors for pointers to members, 6
- addressof
  - Utilities, 219
- adjacent\_difference
  - std, 365
- adjacent\_difference\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 669
- adjacent\_find
  - Non-Mutating, 96
- adjustfield
  - std::basic\_ios, 1120
- std::ios\_base, 1415
- advance
  - std, 365
- algo.h, 1866
- algbase.h, 1874
- algorithmfwd.h, 1875, 1879
- Algorithms, 7
- all
  - std::locale, 1456
- all\_of
  - Non-Mutating, 97
- alloc\_traits.h, 1886, 1887
- allocate
  - \_\_gnu\_cxx::\_alloc\_traits, 435
  - std::allocator\_traits, 1091
- allocate\_shared
  - Pointer\_abstractions, 125
  - std::shared\_ptr, 1707
- allocator.h, 1887
- allocator\_type
  - \_\_gnu\_pbds::trie\_prefix\_search\_node\_update, 929
  - std::allocator\_traits, 1090
  - std::set, 1685
  - std::unordered\_map, 1761
  - std::unordered\_multimap, 1781
  - std::unordered\_multiset, 1802
  - std::unordered\_set, 1822
- Allocators, 8
  - \_\_allocator\_base, 8
- alpha
  - std::gamma\_distribution, 1369
- any\_of
  - Non-Mutating, 97
- app
  - std::basic\_ios, 1120
  - std::ios\_base, 1415
- append
  - \_\_gnu\_cxx::\_versa\_string, 458–460
  - std::basic\_string, 1138–1140
- apply\_generator
  - \_\_gnu\_cxx::typelist, 240
- argument\_type
  - \_\_gnu\_cxx::\_detail::\_Ffit\_finder, 440
  - \_\_gnu\_parallel::\_binder1st, 589
  - \_\_gnu\_parallel::\_binder2nd, 591
  - \_\_gnu\_parallel::\_unary\_negate, 619
  - std::binder1st, 1184
  - std::binder2nd, 1185
  - std::const\_mem\_fun\_ref\_t, 1235
  - std::const\_mem\_fun\_t, 1236
  - std::hash< \_\_gnu\_cxx::throw\_value\_limit >, 1381
  - std::hash< \_\_gnu\_cxx::throw\_value\_random >, 1382
  - std::logical\_not, 1462

- std::mem\_fun\_ref\_t, 1499
- std::mem\_fun\_t, 1501
- std::negate, 1579
- std::pointer\_to\_unary\_function, 1649
- std::unary\_function, 1749
- std::unary\_negate, 1750
- Arithmetic Classes, 10
- array\_allocator.h, 1887
- assign
  - \_\_gnu\_cxx::\_\_versa\_string, 460, 461, 463, 465
  - std::\_\_detail::\_\_Nfa, 1035, 1036
  - std::basic\_regex, 1127–1129
  - std::basic\_string, 1140, 1142, 1143
  - std::deque, 1312, 1313
  - std::forward\_list, 1351, 1352
  - std::list, 1439
  - std::vector, 1846, 1847
- assoc\_container.hpp, 1888
- Associative, 11
- at
  - \_\_gnu\_cxx::\_\_versa\_string, 465
  - std::\_\_detail::\_\_Nfa, 1036
  - std::basic\_string, 1143, 1144
  - std::deque, 1313
  - std::map, 1471
  - std::unordered\_map, 1764, 1765
  - std::vector, 1847, 1848
- ate
  - std::basic\_ios, 1120
  - std::ios\_base, 1415
- atomic\_base.h, 1889
- atomic\_char
  - Atomics, 14
- atomic\_char16\_t
  - Atomics, 14
- atomic\_char32\_t
  - Atomics, 14
- atomic\_int
  - Atomics, 14
- atomic\_int\_fast16\_t
  - Atomics, 14
- atomic\_int\_fast32\_t
  - Atomics, 14
- atomic\_int\_fast64\_t
  - Atomics, 14
- atomic\_int\_fast8\_t
  - Atomics, 14
- atomic\_int\_least16\_t
  - Atomics, 14
- atomic\_int\_least32\_t
  - Atomics, 14
- atomic\_int\_least64\_t
  - Atomics, 14
- atomic\_int\_least8\_t
  - Atomics, 14
- Atomics, 15
- atomic\_intmax\_t
  - Atomics, 15
- atomic\_intptr\_t
  - Atomics, 15
- atomic\_llong
  - Atomics, 15
- atomic\_lockfree\_defines.h, 1890
- atomic\_long
  - Atomics, 15
- atomic\_ptrdiff\_t
  - Atomics, 15
- atomic\_schar
  - Atomics, 15
- atomic\_short
  - Atomics, 15
- atomic\_size\_t
  - Atomics, 15
- atomic\_uchar
  - Atomics, 15
- atomic\_uint
  - Atomics, 16
- atomic\_uint\_fast16\_t
  - Atomics, 16
- atomic\_uint\_fast32\_t
  - Atomics, 16
- atomic\_uint\_fast64\_t
  - Atomics, 16
- atomic\_uint\_fast8\_t
  - Atomics, 16
- atomic\_uint\_least16\_t
  - Atomics, 16
- atomic\_uint\_least32\_t
  - Atomics, 16
- atomic\_uint\_least64\_t
  - Atomics, 16
- atomic\_uint\_least8\_t
  - Atomics, 16
- atomic\_uintmax\_t
  - Atomics, 16
- atomic\_uintptr\_t
  - Atomics, 17
- atomic\_ullong
  - Atomics, 17
- atomic\_ulong
  - Atomics, 17
- atomic\_ushort
  - Atomics, 17
- atomic\_wchar\_t
  - Atomics, 17
- atomic\_word.h, 1891
- atomicity.h, 1891
- Atomics, 12
- ATOMIC\_BOOL\_LOCK\_FREE, 13

- atomic\_char, [14](#)
- atomic\_char16\_t, [14](#)
- atomic\_char32\_t, [14](#)
- atomic\_int, [14](#)
- atomic\_int\_fast16\_t, [14](#)
- atomic\_int\_fast32\_t, [14](#)
- atomic\_int\_fast64\_t, [14](#)
- atomic\_int\_fast8\_t, [14](#)
- atomic\_int\_least16\_t, [14](#)
- atomic\_int\_least32\_t, [14](#)
- atomic\_int\_least64\_t, [14](#)
- atomic\_int\_least8\_t, [15](#)
- atomic\_intmax\_t, [15](#)
- atomic\_intptr\_t, [15](#)
- atomic\_llong, [15](#)
- atomic\_long, [15](#)
- atomic\_ptrdiff\_t, [15](#)
- atomic\_schar, [15](#)
- atomic\_short, [15](#)
- atomic\_size\_t, [15](#)
- atomic\_uchar, [15](#)
- atomic\_uint, [16](#)
- atomic\_uint\_fast16\_t, [16](#)
- atomic\_uint\_fast32\_t, [16](#)
- atomic\_uint\_fast64\_t, [16](#)
- atomic\_uint\_fast8\_t, [16](#)
- atomic\_uint\_least16\_t, [16](#)
- atomic\_uint\_least32\_t, [16](#)
- atomic\_uint\_least64\_t, [16](#)
- atomic\_uint\_least8\_t, [16](#)
- atomic\_uintmax\_t, [16](#)
- atomic\_uintptr\_t, [17](#)
- atomic\_ullong, [17](#)
- atomic\_ulong, [17](#)
- atomic\_ushort, [17](#)
- atomic\_wchar\_t, [17](#)
- kill\_dependency, [17](#)
- memory\_order, [17](#)
- auto\_ptr
  - std::auto\_ptr, [1096](#), [1097](#)
- auto\_ptr.h, [1892](#)
- awk
  - Regular Expressions, [175](#)
- b
  - std::extreme\_value\_distribution, [1339](#)
  - std::weibull\_distribution, [1864](#)
- back
  - \_\_gnu\_cxx::\_\_versa\_string, [467](#)
  - std::\_\_detail::\_\_Nfa, [1037](#)
  - std::basic\_string, [1144](#)
  - std::deque, [1314](#)
  - std::list, [1439](#)
  - std::queue, [1659](#)
  - std::vector, [1848](#)
- back\_insert\_iterator
  - std::back\_insert\_iterator, [1101](#)
- back\_inserter
  - Iterators, [66](#)
- backward\_warning.h, [1892](#)
- bad
  - std::basic\_ios, [1110](#)
- badbit
  - std::basic\_ios, [1120](#)
  - std::ios\_base, [1415](#)
- balanced\_quicksort.h, [1892](#)
- base
  - \_\_gnu\_debug::\_\_Safe\_iterator, [553](#)
  - \_\_gnu\_debug::\_\_Safe\_local\_iterator, [565](#)
  - std::discard\_block\_engine, [1325](#)
  - std::independent\_bits\_engine, [1398](#)
  - std::reverse\_iterator, [1679](#)
  - std::shuffle\_order\_engine, [1712](#)
- Base and Implementation Classes, [18](#), [21](#)
  - \_AnyMatcher, [23](#)
  - \_AutomatonPtr, [22](#)
  - \_Matcher, [22](#)
  - \_Opcode, [23](#)
  - \_S\_invalid\_state\_id, [23](#)
  - \_StateIdT, [22](#)
  - \_StateSet, [22](#)
  - \_StateStack, [23](#)
  - \_Tagger, [23](#)
  - \_TokenT, [23](#)
- Base and Policy Classes, [24–26](#)
- base.h, [1893](#)
- basefield
  - std::basic\_ios, [1120](#)
  - std::ios\_base, [1415](#)
- basic
  - Regular Expressions, [175](#)
- basic\_file.h, [1894](#)
- basic\_ios
  - std::basic\_ios, [1109](#)
- basic\_ios.h, [1894](#)
- basic\_iterator.h, [1895](#)
- basic\_regex
  - std::basic\_regex, [1125](#), [1126](#)
- basic\_string
  - std::basic\_string, [1136–1138](#)
- basic\_string.h, [1895](#)
- before\_begin
  - std::forward\_list, [1352](#)
- beg
  - std::basic\_ios, [1120](#)
  - std::ios\_base, [1416](#)
- begin
  - \_\_gnu\_cxx::\_\_versa\_string, [467](#)

- `__gnu_parallel::PseudoSequence`, 662
  - `__gnu_pbds::sample_trie_access_traits`, 911
  - `__gnu_pbds::trie_string_access_traits`, 932
- `std`, 367
- `std::__detail::Nfa`, 1037
- `std::Temporary_buffer`, 1085
- `std::basic_string`, 1144
- `std::deque`, 1314
- `std::forward_list`, 1352, 1353
- `std::list`, 1440
- `std::map`, 1473
- `std::match_results`, 1491
- `std::multimap`, 1546
- `std::multiset`, 1564
- `std::set`, 1689
- `std::unordered_map`, 1765, 1766
- `std::unordered_multimap`, 1784, 1785
- `std::unordered_multiset`, 1807
- `std::unordered_set`, 1825, 1826
- `std::vector`, 1848
- Bernoulli Distributions, 27
  - `operator!=`, 28
  - `operator<<`, 28
  - `operator>>`, 29
- `bernoulli_distribution`
  - `std::bernoulli_distribution`, 1177
- beta
  - `std::gamma_distribution`, 1369
- `bin_search_tree.hpp`, 1897
- binary
  - `std::basic_ios`, 1120
  - `std::ios_base`, 1416
- Binary Search, 30
  - `binary_search`, 31
  - `equal_range`, 31, 32
  - `lower_bound`, 32, 33
  - `upper_bound`, 33
- `binary_heap.hpp`, 1897
- `binary_heap_const_iterator_`
  - `__gnu_pbds::detail::binary_heap_const_iterator_`, 737
- `binary_heap_point_const_iterator_`
  - `__gnu_pbds::detail::binary_heap_point_const_iterator_`, 740
- `binary_search`
  - Binary Search, 31
- `bind1st`
  - Binder Classes, 35
- `bind2nd`
  - Binder Classes, 36
- Binder Classes, 35
  - `bind1st`, 35
  - `bind2nd`, 36
- `binders.h`, 1898
- `binomial_heap.hpp`, 1898
- `binomial_heap_base.hpp`, 1898
- `bitmap_allocator.h`, 1899
  - `_BALLOC_ALIGN_BYTES`, 1900
- boolalpha
  - `std`, 367
  - `std::basic_ios`, 1120
  - `std::ios_base`, 1416
- Boolean Operations Classes, 37
- `boost_concept_check.h`, 1900
- Branch-Based, 38
- `branch_policy.hpp`, 1901
- bucket
  - `__gnu_debug::Safe_local_iterator`, 565
- `bucket_count`
  - `std::unordered_map`, 1766
  - `std::unordered_multimap`, 1785
  - `std::unordered_multiset`, 1807
  - `std::unordered_set`, 1826
- c
  - `std::queue`, 1660
- `c++0x_warning.h`, 1901
- `c++allocator.h`, 1901
- `c++config.h`, 1901
- `c++io.h`, 1906
- `c++locale.h`, 1906
- `c++locale_internal.h`, 1907
- `c_str`
  - `__gnu_cxx::__versa_string`, 467
  - `std::basic_string`, 1144
- `cache_line_size`
  - `__gnu_parallel::Settings`, 669
- capacity
  - `__gnu_cxx::__versa_string`, 468
  - `std::__detail::Nfa`, 1037
  - `std::basic_string`, 1145
  - `std::vector`, 1849
- `cast.h`, 1907
- category
  - `std::locale`, 1453
- `cbefore_begin`
  - `std::forward_list`, 1353
- `cbegin`
  - `__gnu_cxx::__versa_string`, 468
  - `std::__detail::Nfa`, 1037
  - `std::basic_string`, 1145
  - `std::deque`, 1314
  - `std::forward_list`, 1353
  - `std::list`, 1440
  - `std::map`, 1473
  - `std::match_results`, 1491
  - `std::multimap`, 1547
  - `std::multiset`, 1564

- std::set, 1689
- std::unordered\_map, 1766
- std::unordered\_multimap, 1785
- std::unordered\_multiset, 1808
- std::unordered\_set, 1826, 1827
- std::vector, 1849
- cc\_hash\_max\_collision\_check\_resize\_trigger
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_↔  
resize\_trigger, 706
- cc\_hash\_max\_collision\_check\_resize\_trigger\_imp.hpp, 1908
- cc\_hash\_table
  - \_\_gnu\_pbds::cc\_hash\_table, 711–713
- cc\_ht\_map.hpp, 1908
- cend
  - \_\_gnu\_cxx::\_\_versa\_string, 468
  - std::\_\_detail::\_\_Nfa, 1037
  - std::basic\_string, 1145
  - std::deque, 1314
  - std::forward\_list, 1353
  - std::list, 1440
  - std::map, 1473
  - std::match\_results, 1492
  - std::multimap, 1547
  - std::multiset, 1564
  - std::set, 1689
  - std::unordered\_map, 1766, 1767
  - std::unordered\_multimap, 1787
  - std::unordered\_multiset, 1808
  - std::unordered\_set, 1827
  - std::vector, 1849
- char\_traits.h, 1908
- char\_type
  - std::\_\_ctype\_abstract\_base, 974
  - std::basic\_ios, 1106
  - std::collate, 1223
  - std::collate\_byname, 1229
  - std::ctype< char >, 1251
  - std::ctype< wchar\_t >, 1263
  - std::ctype\_byname< char >, 1291
  - std::istreambuf\_iterator, 1422
  - std::messages, 1507
  - std::money\_get, 1515
  - std::money\_put, 1519
  - std::moneypunct, 1525
  - std::num\_get, 1591
  - std::num\_put, 1605
  - std::numpunct, 1616
  - std::ostream\_iterator, 1628
  - std::ostreambuf\_iterator, 1631
  - std::time\_get, 1727
  - std::time\_put, 1743
- checkers.h, 1909
- classic
  - std::locale, 1454
- classic\_table
  - std::ctype< char >, 1252
  - std::ctype\_byname< char >, 1291
- clear
  - \_\_gnu\_cxx::\_\_versa\_string, 468
  - std::\_\_detail::\_\_Nfa, 1038
  - std::basic\_ios, 1110
  - std::basic\_string, 1145
  - std::deque, 1315
  - std::forward\_list, 1353
  - std::list, 1440
  - std::map, 1473
  - std::multimap, 1547
  - std::multiset, 1564
  - std::set, 1689
  - std::unordered\_map, 1767
  - std::unordered\_multimap, 1787
  - std::unordered\_multiset, 1809
  - std::unordered\_set, 1827
  - std::vector, 1849
- cmp\_fn
  - \_\_gnu\_pbds::tree, 918
- cmp\_fn\_imps.hpp, 1909
- code
  - std::regex\_error, 1664
- codecvt.h, 1909
- codecvt\_specializations.h, 1910
- collate
  - Regular Expressions, 175
  - std::collate, 1223, 1225
  - std::locale, 1456
- combine
  - std::locale, 1454
- compare
  - \_\_gnu\_cxx::\_\_versa\_string, 468–470
  - std::basic\_string, 1145–1147
  - std::collate, 1225
  - std::collate\_byname, 1229
  - std::sub\_match, 1723
- Comparison Classes, 39
- compatibility.h, 1910
- compiletime\_settings.h, 1911
  - \_GLIBCXX\_ASSERTIONS, 1911
  - \_GLIBCXX\_CALL, 1911
  - \_GLIBCXX\_RANDOM\_SHUFFLE\_CONSIDER\_L1, 1912
  - \_GLIBCXX\_RANDOM\_SHUFFLE\_CONSIDER\_T↔  
LB, 1912
  - \_GLIBCXX\_SCALE\_DOWN\_FPU, 1912
  - \_GLIBCXX\_VERBOSE\_LEVEL, 1912
- complex.h, 1912
- concept\_check.h, 1912
- concurrency.h, 1913

- Concurrency, [40](#)
- cond\_dealtor.hpp, [1913](#)
- cond\_key\_dtor\_entry\_dealtor.hpp, [1914](#)
- const\_iterator
  - \_\_gnu\_pbds::trie\_string\_access\_traits, [932](#)
  - std::set, [1685](#)
  - std::unordered\_map, [1761](#)
  - std::unordered\_multimap, [1781](#)
  - std::unordered\_multiset, [1802](#)
  - std::unordered\_set, [1822](#)
- const\_iterator.hpp, [1914](#), [1915](#)
- const\_iterator\_, [952](#)
  - const\_iterator\_, [954](#)
  - const\_pointer, [953](#)
  - const\_reference, [953](#)
  - difference\_type, [953](#)
  - iterator\_category, [953](#)
  - m\_p\_tbl, [955](#)
  - operator!=, [954](#)
  - operator\*, [954](#)
  - operator++, [954](#)
  - operator->, [954](#)
  - operator==, [955](#)
  - pointer, [953](#)
  - reference, [953](#)
  - value\_type, [953](#)
- const\_local\_iterator
  - std::unordered\_map, [1761](#)
  - std::unordered\_multimap, [1781](#)
  - std::unordered\_multiset, [1803](#)
  - std::unordered\_set, [1822](#)
- const\_pointer
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_, [736](#)
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_, [739](#)
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator\_, [781](#)
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_, [785](#)
  - const\_iterator\_, [953](#)
  - iterator\_, [957](#)
  - point\_const\_iterator\_, [960](#)
  - point\_iterator\_, [963](#)
  - std::allocator\_traits, [1090](#)
  - std::set, [1685](#)
  - std::unordered\_map, [1761](#)
  - std::unordered\_multimap, [1781](#)
  - std::unordered\_multiset, [1803](#)
  - std::unordered\_set, [1823](#)
- const\_pointer\_cast
  - std, [368](#)
- const\_reference
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_, [724](#)
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_, [729](#)
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_, [736](#)
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_, [739](#)
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator\_, [781](#)
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_, [785](#)
  - const\_iterator\_, [953](#)
  - iterator\_, [957](#)
  - point\_const\_iterator\_, [960](#)
  - point\_iterator\_, [963](#)
  - std::set, [1685](#)
  - std::unordered\_map, [1761](#)
  - std::unordered\_multimap, [1781](#)
  - std::unordered\_multiset, [1803](#)
  - std::unordered\_set, [1823](#)
- const\_reverse\_iterator
  - std::set, [1685](#)
- const\_void\_pointer
  - \_\_gnu\_cxx::\_\_alloc\_traits, [434](#)
  - std::allocator\_traits, [1090](#)
- construct
  - \_\_gnu\_cxx::\_\_alloc\_traits, [435](#)
  - std::allocator\_traits, [1092](#)
- constructor\_destructor\_fn\_imps.hpp, [1915](#)
- constructor\_destructor\_no\_store\_hash\_fn\_imps.hpp, [1915](#)
- constructor\_destructor\_store\_hash\_fn\_imps.hpp, [1915](#), [1916](#)
- constructors\_destructor\_fn\_imps.hpp, [1916](#), [1917](#)
- container\_base\_dispatch.hpp, [1917](#)
- container\_type
  - std::back\_insert\_iterator, [1101](#)
  - std::front\_insert\_iterator, [1366](#)
  - std::insert\_iterator, [1404](#)
- Containers, [41](#), [42](#)
- copy
  - \_\_gnu\_cxx::\_\_versa\_string, [471](#)
  - Mutating, [72](#)
  - std::basic\_string, [1147](#)
- copy\_backward
  - Mutating, [72](#)
- copy\_exception
  - Exceptions, [46](#)
- copy\_if
  - Mutating, [72](#)
- copy\_n
  - Mutating, [73](#)
- copyfmt
  - std::basic\_ios, [1110](#)



- count
  - Non-Mutating, 97
  - std::map, 1474
  - std::multimap, 1547
  - std::multiset, 1564
  - std::set, 1689
  - std::unordered\_map, 1767
  - std::unordered\_multimap, 1787
  - std::unordered\_multiset, 1809
  - std::unordered\_set, 1827
- count\_if
  - Non-Mutating, 98
- count\_minimal\_n
  - \_\_gnu\_parallel::Settings, 670
- cpp\_type\_traits.h, 1918
- cpu\_defines.h, 1918
- crbegin
  - \_\_gnu\_cxx::\_\_versa\_string, 471
  - std::\_\_detail::\_\_Nfa, 1038
  - std::basic\_string, 1149
  - std::deque, 1315
  - std::list, 1440
  - std::map, 1474
  - std::multimap, 1547
  - std::multiset, 1566
  - std::set, 1691
  - std::vector, 1849
- cregex\_token\_iterator
  - Regular Expressions, 145
- crend
  - \_\_gnu\_cxx::\_\_versa\_string, 471
  - std::\_\_detail::\_\_Nfa, 1038
  - std::basic\_string, 1149
  - std::deque, 1315
  - std::list, 1440
  - std::map, 1474
  - std::multimap, 1548
  - std::multiset, 1566
  - std::set, 1691
  - std::vector, 1849
- csub\_match
  - Regular Expressions, 145
- ctype
  - std::ctype< char >, 1251
  - std::ctype< wchar\_t >, 1263
  - std::locale, 1456
- ctype\_base.h, 1919
- ctype\_inline.h, 1919
- cur
  - std::basic\_ios, 1120
  - std::ios\_base, 1416
- curr\_symbol
  - std::moneypunct, 1526
  - std::moneypunct\_byname, 1534
- current\_exception
  - Exceptions, 47
- cxxabi.h, 1919
- cxxabi\_forced.h, 1921
- cxxabi\_tweaks.h, 1921
- data
  - \_\_gnu\_cxx::\_\_versa\_string, 472
  - std::\_\_detail::\_\_Nfa, 1038
  - std::basic\_string, 1149
  - std::vector, 1849
- Data Structure Type, 43
- date\_order
  - std::time\_get, 1728
  - std::time\_get\_byname, 1736
- deallocate
  - \_\_gnu\_cxx::\_\_alloc\_traits, 435
  - std::allocator\_traits, 1092
- debug.h, 1921
- debug\_allocator.h, 1922
- debug\_fn\_imps.hpp, 1922–1924
- debug\_map\_base.hpp, 1924
- debug\_no\_store\_hash\_fn\_imps.hpp, 1924, 1925
- debug\_store\_hash\_fn\_imps.hpp, 1925
- dec
  - std, 368
  - std::basic\_ios, 1121
  - std::ios\_base, 1416
- decimal\_point
  - std::moneypunct, 1526
  - std::moneypunct\_byname, 1535
  - std::numpunct, 1618
  - std::numpunct\_byname, 1623
- densities
  - std::piecewise\_constant\_distribution, 1638
  - std::piecewise\_linear\_distribution, 1643
- deque
  - std::deque, 1305, 1307, 1309
- destroy
  - \_\_gnu\_cxx::\_\_alloc\_traits, 437
  - std::allocator\_traits, 1092
- Diagnostics, 44
- difference\_type
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_↔  
it\_, 724
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_, 729
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_,  
736
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_↔  
iterator\_, 739
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_↔  
const\_iterator\_, 781
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_↔  
node\_point\_const\_iterator\_, 785

- const\_iterator\_, 953
- iterator\_, 957
- point\_const\_iterator\_, 960
- point\_iterator\_, 963
- std::allocator\_traits, 1090
- std::back\_insert\_iterator, 1101
- std::front\_insert\_iterator, 1366
- std::insert\_iterator, 1404
- std::istream\_iterator, 1420
- std::istreambuf\_iterator, 1422
- std::iterator, 1425
- std::ostream\_iterator, 1628
- std::ostreambuf\_iterator, 1631
- std::pointer\_traits, 1650
- std::pointer\_traits< \_Tp \* >, 1650
- std::raw\_storage\_iterator, 1663
- std::set, 1685
- std::unordered\_map, 1761
- std::unordered\_multimap, 1781
- std::unordered\_multiset, 1803
- std::unordered\_set, 1823
- direct\_mask\_range\_hashing\_imp.hpp, 1925
- direct\_mod\_range\_hashing\_imp.hpp, 1925
- discard
  - std::discard\_block\_engine, 1325
  - std::independent\_bits\_engine, 1399
  - std::linear\_congruential\_engine, 1431
  - std::mersenne\_twister\_engine, 1503
  - std::shuffle\_order\_engine, 1712
- discard\_block\_engine
  - std::discard\_block\_engine, 1324, 1325
- distance
  - std, 368
- do\_compare
  - std::collate, 1225
  - std::collate\_byname, 1230
- do\_curr\_symbol
  - std::moneypunct, 1527
  - std::moneypunct\_byname, 1535
- do\_date\_order
  - std::time\_get, 1728
  - std::time\_get\_byname, 1736
- do\_decimal\_point
  - std::moneypunct, 1527
  - std::moneypunct\_byname, 1535
  - std::numpunct, 1618
  - std::numpunct\_byname, 1623
- do\_falsename
  - std::numpunct, 1619
  - std::numpunct\_byname, 1624
- do\_frac\_digits
  - std::moneypunct, 1527
  - std::moneypunct\_byname, 1535
- do\_get
  - std::money\_get, 1516
  - std::num\_get, 1591–1596
- do\_get\_date
  - std::time\_get, 1729
  - std::time\_get\_byname, 1736
- do\_get\_monthname
  - std::time\_get, 1729
  - std::time\_get\_byname, 1737
- do\_get\_time
  - std::time\_get, 1730
  - std::time\_get\_byname, 1737
- do\_get\_weekday
  - std::time\_get, 1730
  - std::time\_get\_byname, 1738
- do\_get\_year
  - std::time\_get, 1731
  - std::time\_get\_byname, 1738
- do\_grouping
  - std::moneypunct, 1527
  - std::moneypunct\_byname, 1536
  - std::numpunct, 1619
  - std::numpunct\_byname, 1624
- do\_hash
  - std::collate, 1226
  - std::collate\_byname, 1230
- do\_is
  - std::\_\_ctype\_abstract\_base, 974
  - std::ctype, 1239
  - std::ctype< wchar\_t >, 1263, 1265
  - std::ctype\_byname, 1279
- do\_narrow
  - std::\_\_ctype\_abstract\_base, 974, 975
  - std::ctype, 1239, 1240
  - std::ctype< char >, 1252
  - std::ctype< wchar\_t >, 1265
  - std::ctype\_byname, 1279, 1280
  - std::ctype\_byname< char >, 1291
- do\_neg\_format
  - std::moneypunct, 1528
  - std::moneypunct\_byname, 1536
- do\_negative\_sign
  - std::moneypunct, 1528
  - std::moneypunct\_byname, 1536
- do\_out
  - std::\_\_codecvt\_abstract\_base, 969
  - std::codecvt, 1202
  - std::codecvt< \_InternT, \_ExternT, encoding\_state >, 1206
  - std::codecvt< char, char, mbstate\_t >, 1210
  - std::codecvt< wchar\_t, char, mbstate\_t >, 1214
  - std::codecvt\_byname, 1219
- do\_pos\_format
  - std::moneypunct, 1528
  - std::moneypunct\_byname, 1537

- do\_positive\_sign
  - std::moneypunct, 1529
  - std::moneypunct\_byname, 1537
- do\_put
  - std::money\_put, 1520
  - std::num\_put, 1605, 1607–1609
  - std::time\_put, 1744
  - std::time\_put\_byname, 1746
- do\_scan\_is
  - std::\_\_ctype\_abstract\_base, 975
  - std::ctype, 1240
  - std::ctype< wchar\_t >, 1266
  - std::ctype\_byname, 1280
- do\_scan\_not
  - std::\_\_ctype\_abstract\_base, 976
  - std::ctype, 1241
  - std::ctype< wchar\_t >, 1266
  - std::ctype\_byname, 1281
- do\_thousands\_sep
  - std::moneypunct, 1529
  - std::moneypunct\_byname, 1537
  - std::numpunct, 1619
  - std::numpunct\_byname, 1624
- do\_tolower
  - std::\_\_ctype\_abstract\_base, 976
  - std::ctype, 1241
  - std::ctype< char >, 1252, 1253
  - std::ctype< wchar\_t >, 1267
  - std::ctype\_byname, 1281
  - std::ctype\_byname< char >, 1292
- do\_toupper
  - std::\_\_ctype\_abstract\_base, 977
  - std::ctype, 1242
  - std::ctype< char >, 1253
  - std::ctype< wchar\_t >, 1267, 1269
  - std::ctype\_byname, 1283
  - std::ctype\_byname< char >, 1292, 1294
- do\_transform
  - std::collate, 1226
  - std::collate\_byname, 1230
- do\_truename
  - std::numpunct, 1619
  - std::numpunct\_byname, 1624
- do\_widen
  - std::\_\_ctype\_abstract\_base, 978
  - std::ctype, 1242, 1243
  - std::ctype< char >, 1254
  - std::ctype< wchar\_t >, 1269
  - std::ctype\_byname, 1283, 1284
  - std::ctype\_byname< char >, 1294
- dynamic\_pointer\_cast
  - std, 368
- e\_pos
  - \_\_gnu\_pbds::sample\_trie\_access\_traits, 911
  - \_\_gnu\_pbds::trie\_string\_access\_traits, 932
- e\_type
  - \_\_gnu\_pbds::sample\_trie\_access\_traits, 911
  - \_\_gnu\_pbds::trie\_string\_access\_traits, 932
- ECMAScript
  - Regular Expressions, 175
- egrep
  - Regular Expressions, 175
- element\_type
  - std::auto\_ptr, 1096
  - std::pointer\_traits, 1650
  - std::pointer\_traits< \_Tp \* >, 1650
- emplace
  - std::\_\_detail::\_\_Nfa, 1038
  - std::deque, 1315
  - std::list, 1441
  - std::map, 1474
  - std::multimap, 1548
  - std::multiset, 1566
  - std::set, 1691
  - std::unordered\_map, 1767
  - std::unordered\_multimap, 1787
  - std::unordered\_multiset, 1809
  - std::unordered\_set, 1829
  - std::vector, 1850
- emplace\_after
  - std::forward\_list, 1353
- emplace\_front
  - std::forward\_list, 1354
- emplace\_hint
  - std::map, 1475
  - std::multimap, 1548
  - std::multiset, 1566
  - std::set, 1691
  - std::unordered\_map, 1768
  - std::unordered\_multimap, 1789
  - std::unordered\_multiset, 1809
  - std::unordered\_set, 1829
- empty
  - \_\_gnu\_cxx::\_\_versa\_string, 472
  - \_\_gnu\_pbds::detail::cc\_ht\_map, 751
  - \_\_gnu\_pbds::detail::gp\_ht\_map, 774
  - std::\_\_detail::\_\_Nfa, 1039
  - std::basic\_string, 1149
  - std::deque, 1315
  - std::forward\_list, 1354
  - std::list, 1441
  - std::map, 1475
  - std::match\_results, 1492
  - std::multimap, 1548
  - std::multiset, 1567
  - std::priority\_queue, 1657
  - std::queue, 1659

- std::set, 1692
- std::stack, 1717
- std::unordered\_map, 1768
- std::unordered\_multimap, 1789
- std::unordered\_multiset, 1810
- std::unordered\_set, 1829
- std::vector, 1850
- enc\_filebuf.h, 1925
- end
  - \_\_gnu\_cxx::\_\_versa\_string, 472
  - \_\_gnu\_parallel::\_PseudoSequence, 662
  - \_\_gnu\_pbds::sample\_trie\_access\_traits, 912
  - \_\_gnu\_pbds::trie\_string\_access\_traits, 932
  - std, 368, 369
  - std::\_\_detail::\_Nfa, 1039
  - std::\_Temporary\_buffer, 1085
  - std::basic\_ios, 1121
  - std::basic\_string, 1149
  - std::deque, 1315, 1316
  - std::forward\_list, 1354
  - std::ios\_base, 1416
  - std::list, 1441
  - std::map, 1475
  - std::match\_results, 1492
  - std::multimap, 1549
  - std::multiset, 1567
  - std::set, 1692
  - std::unordered\_map, 1768, 1769
  - std::unordered\_multimap, 1789, 1790
  - std::unordered\_multiset, 1810
  - std::unordered\_set, 1830
  - std::vector, 1850
- entry\_cmp.hpp, 1926
- entry\_list\_fn\_imps.hpp, 1926
- entry\_metadata\_base.hpp, 1926
- entry\_pred.hpp, 1926
- eof
  - std::basic\_ios, 1110
- eofbit
  - std::basic\_ios, 1121
  - std::ios\_base, 1416
- eq\_by\_less.hpp, 1927
- equal
  - Non-Mutating, 98
  - std::istreambuf\_iterator, 1424
- equal\_range
  - Binary Search, 31, 32
  - std::map, 1475, 1477
  - std::multimap, 1549
  - std::multiset, 1567
  - std::set, 1692
  - std::unordered\_map, 1769
  - std::unordered\_multimap, 1790
  - std::unordered\_multiset, 1812
- std::unordered\_set, 1831, 1832
- equally\_split.h, 1927
- erase
  - \_\_gnu\_cxx::\_\_versa\_string, 472, 473
  - std::\_\_detail::\_Nfa, 1039
  - std::basic\_string, 1150
  - std::deque, 1316
  - std::list, 1441, 1442
  - std::map, 1477, 1478
  - std::multimap, 1550
  - std::multiset, 1568
  - std::set, 1694
  - std::unordered\_map, 1771, 1772
  - std::unordered\_multimap, 1792, 1793
  - std::unordered\_multiset, 1812, 1813
  - std::unordered\_set, 1832, 1833
  - std::vector, 1850, 1852
- erase\_after
  - std::forward\_list, 1354, 1355
- erase\_can\_throw
  - Traits, 213
- erase\_fn\_imps.hpp, 1927–1929
- erase\_no\_store\_hash\_fn\_imps.hpp, 1929
- erase\_store\_hash\_fn\_imps.hpp, 1929
- error\_backref
  - Regular Expressions, 146
- error\_badbrace
  - Regular Expressions, 146
- error\_badrepeat
  - Regular Expressions, 147
- error\_brace
  - Regular Expressions, 147
- error\_brack
  - Regular Expressions, 147
- error\_collate
  - Regular Expressions, 147
- error\_complexity
  - Regular Expressions, 147
- error\_constants.h, 1929
- error\_ctype
  - Regular Expressions, 147
- error\_escape
  - Regular Expressions, 147
- error\_paren
  - Regular Expressions, 147
- error\_range
  - Regular Expressions, 147
- error\_space
  - Regular Expressions, 147
- error\_stack
  - Regular Expressions, 147
- error\_type
  - Regular Expressions, 146
- event

- std::basic\_ios, 1109
- std::ios\_base, 1410
- event\_callback
  - std::basic\_ios, 1106
  - std::ios\_base, 1408
- exception.hpp, 1930
- exception\_defines.h, 1931
- exception\_ptr.h, 1931
- Exceptions, 45, 46
  - copy\_exception, 46
  - current\_exception, 47
  - make\_exception\_ptr, 47
  - rethrow\_exception, 47
  - rethrow\_if\_nested, 47
  - throw\_with\_nested, 47
- exceptions
  - std::basic\_ios, 1111
- exponential\_distribution
  - std::exponential\_distribution, 1336
- extc++.h, 1931
- extended
  - Regular Expressions, 175
- Extensions, 48
- external\_load\_access
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_↵  
resize\_trigger, 706
  - \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger, 879
- extptr\_allocator.h, 1932
- facet
  - std::locale::facet, 1459
- fail
  - std::basic\_ios, 1111
- failbit
  - std::basic\_ios, 1121
  - std::ios\_base, 1416
- failed
  - std::ostreambuf\_iterator, 1632
- falsename
  - std::numpunct, 1620
  - std::numpunct\_byname, 1625
- fd
  - \_\_gnu\_cxx::stdio\_filebuf, 535
- features.h, 1932
  - \_GLIBCXX\_BAL\_QUICKSORT, 1933
  - \_GLIBCXX\_FIND\_CONSTANT\_SIZE\_BLOCKS,  
1933
  - \_GLIBCXX\_FIND\_EQUAL\_SPLIT, 1933
  - \_GLIBCXX\_FIND\_GROWING\_BLOCKS, 1933
  - \_GLIBCXX\_MERGESORT, 1933
  - \_GLIBCXX\_QUICKSORT, 1933
  - \_GLIBCXX\_TREE\_DYNAMIC\_BALANCING, 1933
  - \_GLIBCXX\_TREE\_FULL\_COPY, 1934
  - \_GLIBCXX\_TREE\_INITIAL\_SPLITTING, 1934
- fenv.h, 1934
- file
  - \_\_gnu\_cxx::stdio\_filebuf, 535
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 537
- fill
  - Mutating, 73
  - std::basic\_ios, 1112
- fill\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 670
- fill\_n
  - Mutating, 73
- find
  - \_\_gnu\_cxx::\_\_versa\_string, 473, 474
  - Non-Mutating, 99
  - std::basic\_string, 1151, 1152
  - std::map, 1478, 1479
  - std::multimap, 1552
  - std::multiset, 1570
  - std::set, 1696
  - std::unordered\_map, 1772
  - std::unordered\_multimap, 1793
  - std::unordered\_multiset, 1814
  - std::unordered\_set, 1833, 1835
- find.h, 1934
- find\_by\_order
  - \_\_gnu\_pbds::tree\_order\_statistics\_node\_update,  
921
  - \_\_gnu\_pbds::trie\_order\_statistics\_node\_update, 926
- find\_end
  - Non-Mutating, 99, 100
- find\_first\_not\_of
  - \_\_gnu\_cxx::\_\_versa\_string, 476, 478
  - std::basic\_string, 1152, 1153
- find\_first\_of
  - \_\_gnu\_cxx::\_\_versa\_string, 478, 480
  - Non-Mutating, 100, 101
  - std::basic\_string, 1153, 1154
- find\_fn\_imps.hpp, 1935, 1936
- find\_if
  - Non-Mutating, 101
- find\_if\_not
  - Non-Mutating, 101
- find\_increasing\_factor
  - \_\_gnu\_parallel::\_Settings, 670
- find\_initial\_block\_size
  - \_\_gnu\_parallel::\_Settings, 670
- find\_last\_not\_of
  - \_\_gnu\_cxx::\_\_versa\_string, 480, 482
  - std::basic\_string, 1155, 1156
- find\_last\_of
  - \_\_gnu\_cxx::\_\_versa\_string, 484, 486
  - std::basic\_string, 1156, 1157
- find\_maximum\_block\_size
  - \_\_gnu\_parallel::\_Settings, 670

- find\_no\_store\_hash\_fn\_imps.hpp, 1936
- find\_scale\_factor
  - \_\_gnu\_parallel::Settings, 670
- find\_selectors.h, 1936
- find\_sequential\_search\_size
  - \_\_gnu\_parallel::Settings, 670
- find\_store\_hash\_fn\_imps.hpp, 1937
- first
  - \_\_gnu\_parallel::IteratorPair, 629
  - std::pair, 1637
  - std::sub\_match, 1724
- first\_argument\_type
  - \_\_gnu\_parallel::EqualFromLess, 624
  - \_\_gnu\_parallel::EqualTo, 625
  - \_\_gnu\_parallel::Less, 632
  - \_\_gnu\_parallel::Lexicographic, 634
  - \_\_gnu\_parallel::LexicographicReverse, 635
  - \_\_gnu\_parallel::Multiplies, 656
  - \_\_gnu\_parallel::Plus, 659
  - std::binary\_function, 1181
  - std::binary\_negate, 1183
  - std::const\_mem\_fun1\_ref\_t, 1232
  - std::const\_mem\_fun1\_t, 1234
  - std::divides, 1333
  - std::equal\_to, 1334
  - std::greater, 1375
  - std::greater\_equal, 1376
  - std::less, 1427
  - std::less\_equal, 1428
  - std::logical\_and, 1461
  - std::logical\_or, 1463
  - std::mem\_fun1\_ref\_t, 1497
  - std::mem\_fun1\_t, 1498
  - std::minus, 1511
  - std::modulus, 1512
  - std::multiplies, 1560
  - std::not\_equal\_to, 1588
  - std::owner\_less< shared\_ptr< \_Tp > >, 1634
  - std::owner\_less< weak\_ptr< \_Tp > >, 1635
  - std::plus, 1646
  - std::pointer\_to\_binary\_function, 1647
- fixed
  - std, 369
  - std::basic\_ios, 1121
  - std::ios\_base, 1416
- flags
  - std::basic\_ios, 1112
  - std::basic\_regex, 1129
  - std::ios\_base, 1411
- floatfield
  - std::basic\_ios, 1121
  - std::ios\_base, 1417
- fmtflags
  - std::basic\_ios, 1107
- std::ios\_base, 1408
- for\_each
  - Non-Mutating, 102
- for\_each.h, 1937
- for\_each\_minimal\_n
  - \_\_gnu\_parallel::Settings, 670
- for\_each\_selectors.h, 1937
- format
  - std::match\_results, 1492, 1493
- format\_default
  - Regular Expressions, 176
- format\_first\_only
  - Regular Expressions, 176
- format\_no\_copy
  - Regular Expressions, 176
- format\_sed
  - Regular Expressions, 176
- formatter.h, 1938
- forward
  - Utilities, 219
- forward\_list
  - std::forward\_list, 1349–1351
- forward\_list.h, 1939
- fpos
  - std::fpos, 1364
- frac\_digits
  - std::moneypunct, 1529
  - std::moneypunct\_byname, 1538
- front
  - \_\_gnu\_cxx::\_\_versa\_string, 486
  - std::\_\_detail::\_\_Nfa, 1040
  - std::basic\_string, 1157
  - std::deque, 1316, 1317
  - std::forward\_list, 1355
  - std::list, 1442
  - std::queue, 1659
  - std::vector, 1852
- front\_insert\_iterator
  - std::front\_insert\_iterator, 1366
- front\_inserter
  - Iterators, 66
- functexcept.h, 1940
- Function Objects, 49
- functional\_hash.h, 1940
- functions.h, 1941
- gamma\_distribution
  - std::gamma\_distribution, 1369
- generate
  - Mutating, 75
- generate\_canonical
  - Random Number Generation, 133
- generate\_minimal\_n
  - \_\_gnu\_parallel::Settings, 670

generate\_n  
     Mutating, 75  
 get  
     \_\_gnu\_parallel::\_\_Settings, 669  
     std::auto\_ptr, 1097  
     std::money\_get, 1516, 1517  
     std::num\_get, 1597–1602  
 get\_actual\_size  
     \_\_gnu\_pbds::hash\_standard\_resize\_policy, 883  
 get\_allocator  
     \_\_gnu\_cxx::\_\_versa\_string, 486  
     std::basic\_string, 1158  
     std::deque, 1317  
     std::forward\_list, 1355  
     std::list, 1442  
     std::map, 1479  
     std::match\_results, 1493  
     std::multimap, 1552  
     std::multiset, 1570  
     std::set, 1696  
     std::unordered\_map, 1773  
     std::unordered\_multimap, 1794  
     std::unordered\_multiset, 1814  
     std::unordered\_set, 1835  
 get\_child  
     \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer, 817  
     \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_iter, 820  
 get\_comb\_hash\_fn  
     \_\_gnu\_pbds::detail::cc\_ht\_map, 751  
 get\_comb\_probe\_fn  
     \_\_gnu\_pbds::detail::gp\_ht\_map, 774  
 get\_date  
     std::time\_get, 1731  
     std::time\_get\_byname, 1739  
 get\_deleter  
     Pointer\_abstractions, 125  
 get\_eq\_fn  
     \_\_gnu\_pbds::detail::cc\_ht\_map, 751  
     \_\_gnu\_pbds::detail::gp\_ht\_map, 774  
 get\_hash\_fn  
     \_\_gnu\_pbds::detail::cc\_ht\_map, 752  
     \_\_gnu\_pbds::detail::gp\_ht\_map, 774  
 get\_l\_child  
     \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_↵  
         it\_, 725  
     \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_, 730  
     \_\_gnu\_pbds::detail::ov\_tree\_node\_const\_it\_, 798  
     \_\_gnu\_pbds::detail::ov\_tree\_node\_it\_, 799  
 get\_load  
     \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_↵  
         resize\_trigger, 707  
 get\_loads  
     \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger, 880  
 get\_metadata  
     \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_↵  
         it\_, 725  
     \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_, 730  
     \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer, 817  
     \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_iter, 820  
 get\_monthname  
     std::time\_get, 1732  
     std::time\_get\_byname, 1739  
 get\_nearest\_larger\_size  
     \_\_gnu\_pbds::sample\_size\_policy, 910  
 get\_nearest\_smaller\_size  
     \_\_gnu\_pbds::sample\_size\_policy, 910  
 get\_new\_size  
     \_\_gnu\_pbds::hash\_standard\_resize\_policy, 883  
     \_\_gnu\_pbds::sample\_resize\_policy, 906  
 get\_probe\_fn  
     \_\_gnu\_pbds::detail::gp\_ht\_map, 775  
 get\_r\_child  
     \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_↵  
         it\_, 725  
     \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_, 730  
     \_\_gnu\_pbds::detail::ov\_tree\_node\_const\_it\_, 798  
     \_\_gnu\_pbds::detail::ov\_tree\_node\_it\_, 799  
 get\_resize\_policy  
     \_\_gnu\_pbds::detail::cc\_ht\_map, 752  
     \_\_gnu\_pbds::detail::gp\_ht\_map, 775  
 get\_size\_policy  
     \_\_gnu\_pbds::hash\_standard\_resize\_policy, 883  
 get\_temporary\_buffer  
     std, 369  
 get\_time  
     std::time\_get, 1732  
     std::time\_get\_byname, 1740  
 get\_trigger\_policy  
     \_\_gnu\_pbds::hash\_standard\_resize\_policy, 883  
 get\_weekday  
     std::time\_get, 1733  
     std::time\_get\_byname, 1740  
 get\_year  
     std::time\_get, 1733  
     std::time\_get\_byname, 1741  
 getline  
     std, 369, 371  
 getloc  
     std::basic\_ios, 1113  
     std::basic\_regex, 1129  
     std::ios\_base, 1411  
     std::regex\_traits, 1673  
 global  
     std::locale, 1455  
 good  
     std::basic\_ios, 1113  
 goodbit  
     std::basic\_ios, 1121



- std::ios\_base, 1417
- gp\_hash\_table
  - \_\_gnu\_pbds::gp\_hash\_table, 873–876
- gp\_ht\_map.hpp, 1943
- grep
  - Regular Expressions, 176
- grouping
  - std::moneypunct, 1530
  - std::moneypunct\_byname, 1538
  - std::numpunct, 1620
  - std::numpunct\_byname, 1625
- gslice
  - Numeric\_arrays, 114
- gslice.h, 1943
- gslice\_array
  - Numeric\_arrays, 114
- gslice\_array.h, 1944
- hash
  - std::collate, 1226
  - std::collate\_byname, 1230
- Hash-Based, 51
- hash\_bytes.h, 1944
- hash\_eq\_fn.hpp, 1945
- hash\_exponential\_size\_policy
  - \_\_gnu\_pbds::hash\_exponential\_size\_policy, 878
- hash\_exponential\_size\_policy\_imp.hpp, 1945
- hash\_fun.h, 1945
- hash\_function
  - std::unordered\_map, 1773
  - std::unordered\_multimap, 1794
  - std::unordered\_multiset, 1814
  - std::unordered\_set, 1835
- hash\_load\_check\_resize\_trigger
  - \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger, 879
- hash\_load\_check\_resize\_trigger\_imp.hpp, 1945
- hash\_load\_check\_resize\_trigger\_size\_base.hpp, 1946
- hash\_policy.hpp, 1946
- hash\_prime\_size\_policy
  - \_\_gnu\_pbds::hash\_prime\_size\_policy, 881
- hash\_prime\_size\_policy\_imp.hpp, 1947
- hash\_standard\_resize\_policy
  - \_\_gnu\_pbds::hash\_standard\_resize\_policy, 882
- hash\_standard\_resize\_policy\_imp.hpp, 1947
- hasher
  - std::unordered\_map, 1762
  - std::unordered\_multimap, 1782
  - std::unordered\_multiset, 1803
  - std::unordered\_set, 1823
- Hashes, 52
- hashtable.h, 1947, 1948
- hashtable\_policy.h, 1948
- Heap, 53
  - is\_heap, 53, 54
  - is\_heap\_until, 54
  - make\_heap, 55
  - pop\_heap, 55, 56
  - push\_heap, 56
  - sort\_heap, 56, 58
- Heap-Based, 59
  - priority\_queue, 60
- hex
  - std, 373
  - std::basic\_ios, 1121
  - std::ios\_base, 1417
- icase
  - Regular Expressions, 176
- id
  - std::collate, 1227
  - std::collate\_byname, 1231
  - std::ctype, 1248
  - std::ctype< char >, 1260
  - std::ctype< wchar\_t >, 1275
  - std::ctype\_byname, 1288
  - std::ctype\_byname< char >, 1299
  - std::locale::id, 1460
  - std::messages, 1507
  - std::messages\_byname, 1510
  - std::money\_get, 1518
  - std::money\_put, 1523
  - std::moneypunct, 1532
  - std::moneypunct\_byname, 1540
  - std::num\_get, 1603
  - std::num\_put, 1614
  - std::numpunct, 1621
  - std::numpunct\_byname, 1626
  - std::time\_get, 1734
  - std::time\_get\_byname, 1741
  - std::time\_put, 1745
  - std::time\_put\_byname, 1748
- imbue
  - std::basic\_ios, 1113
  - std::basic\_regex, 1129
  - std::ios\_base, 1411
  - std::regex\_traits, 1673
- in
  - std::\_\_codecvt\_abstract\_base, 970
  - std::basic\_ios, 1121
  - std::codecvt, 1203
  - std::codecvt< \_InternT, \_ExternT, encoding\_state >, 1206
  - std::codecvt< char, char, mbstate\_t >, 1210
  - std::codecvt< wchar\_t, char, mbstate\_t >, 1215
  - std::codecvt\_byname, 1220
  - std::ios\_base, 1417
- includes
  - Set Operation, 182, 183



- increment
  - std::linear\_congruential\_engine, 1433
- independent\_bits\_engine
  - std::independent\_bits\_engine, 1397, 1398
- indirect\_array
  - Numeric\_arrays, 115
- indirect\_array.h, 1950
- info\_fn\_imps.hpp, 1951, 1952
- init
  - std::basic\_ios, 1113
- inner\_product
  - std, 373
- inplace\_merge
  - Sorting, 191
- insert
  - \_\_gnu\_cxx::\_\_versa\_string, 487–490
  - std::\_\_detail::\_\_Nfa, 1040, 1042
  - std::basic\_string, 1158, 1160–1162
  - std::deque, 1317, 1318
  - std::list, 1442, 1444, 1445
  - std::map, 1479, 1480
  - std::multimap, 1552, 1554
  - std::multiset, 1570, 1572
  - std::set, 1696, 1698
  - std::unordered\_map, 1773–1775
  - std::unordered\_multimap, 1794–1796
  - std::unordered\_multiset, 1815, 1816
  - std::unordered\_set, 1835–1837
  - std::vector, 1852–1854
- insert\_after
  - std::forward\_list, 1355, 1357
- insert\_fn\_imps.hpp, 1952, 1953
- insert\_iterator
  - std::insert\_iterator, 1404
- insert\_join\_fn\_imps.hpp, 1953
- insert\_no\_store\_hash\_fn\_imps.hpp, 1954
- insert\_store\_hash\_fn\_imps.hpp, 1954
- inserter
  - Iterators, 66
- int\_type
  - std::basic\_ios, 1107
  - std::istreambuf\_iterator, 1423
- internal
  - std, 374
  - std::basic\_ios, 1122
  - std::ios\_base, 1417
- intervals
  - std::piecewise\_constant\_distribution, 1639
  - std::piecewise\_linear\_distribution, 1643
- intl
  - std::moneypunct, 1532
- Invalidation Guarantees, 61
- ios\_base.h, 1954
- iostate
  - std::basic\_ios, 1107
  - std::ios\_base, 1409
- iota
  - std, 374
- is
  - std::\_\_ctype\_abstract\_base, 978, 979
  - std::ctype, 1243, 1244
  - std::ctype< char >, 1254, 1256
  - std::ctype< wchar\_t >, 1270
  - std::ctype\_byname, 1284
  - std::ctype\_byname< char >, 1295
- is\_grow\_needed
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_↔  
resize\_trigger, 707
  - \_\_gnu\_pbds::sample\_resize\_trigger, 908
- is\_heap
  - Heap, 53, 54
- is\_heap\_until
  - Heap, 54
- is\_partitioned
  - Mutating, 75
- is\_permutation
  - Non-Mutating, 102
- is\_resize\_needed
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_↔  
resize\_trigger, 707
  - \_\_gnu\_pbds::sample\_resize\_policy, 906
  - \_\_gnu\_pbds::sample\_resize\_trigger, 908
- is\_sorted
  - Sorting, 191, 193
- is\_sorted\_until
  - Sorting, 193
- isalnum
  - std, 374
- isalpha
  - std, 374
- isctrl
  - std, 374
- isctype
  - Regular Expressions, 147
- isdigit
  - std, 374
- isgraph
  - std, 375
- islower
  - std, 375
- isprint
  - std, 375
- ispunct
  - std, 375
- isspace
  - std, 375
- istream\_iterator
  - std::istream\_iterator, 1421

- istream\_type
  - std::istreambuf\_iterator, [1423](#)
- istreambuf\_iterator
  - std::istreambuf\_iterator, [1424](#)
- isupper
  - std, [375](#)
- isxdigit
  - std, [375](#)
- iter\_swap
  - Mutating, [77](#)
- iter\_type
  - std::money\_get, [1515](#)
  - std::money\_put, [1519](#)
  - std::num\_get, [1591](#)
  - std::num\_put, [1605](#)
  - std::time\_get, [1727](#)
  - std::time\_put, [1743](#)
- iterator
  - std::set, [1685](#)
  - std::unordered\_map, [1762](#)
  - std::unordered\_multimap, [1782](#)
  - std::unordered\_multiset, [1803](#)
  - std::unordered\_set, [1823](#)
- Iterator Tags, [62](#)
- iterator.h, [1956](#)
- iterator.hpp, [1956](#)
- iterator\_, [955](#)
  - const\_pointer, [957](#)
  - const\_reference, [957](#)
  - difference\_type, [957](#)
  - iterator\_, [957](#)
  - iterator\_category, [957](#)
  - m\_p\_tbl, [959](#)
  - operator const point\_iterator\_, [958](#)
  - operator point\_iterator\_, [958](#)
  - operator!=, [958](#)
  - operator\*, [958](#)
  - operator++, [958](#)
  - operator->, [958](#)
  - operator==, [958](#)
  - pointer, [957](#)
  - reference, [957](#)
  - value\_type, [957](#)
- iterator\_category
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_↵  
it\_, [724](#)
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_, [729](#)
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_,  
[736](#)
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_↵  
iterator\_, [740](#)
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_↵  
const\_iterator\_, [781](#)
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_↵  
node\_point\_const\_iterator\_, [785](#)
  - const\_iterator\_, [953](#)
  - iterator\_, [957](#)
  - point\_const\_iterator\_, [960](#)
  - point\_iterator\_, [963](#)
  - std::back\_insert\_iterator, [1101](#)
  - std::front\_insert\_iterator, [1366](#)
  - std::insert\_iterator, [1404](#)
  - std::istream\_iterator, [1420](#)
  - std::istreambuf\_iterator, [1423](#)
  - std::iterator, [1425](#)
  - std::ostream\_iterator, [1628](#)
  - std::ostreambuf\_iterator, [1631](#)
  - std::raw\_storage\_iterator, [1663](#)
  - std::reverse\_iterator, [1678](#)
- iterator\_fn\_imps.hpp, [1956](#)
- iterator\_tracker.h, [1956](#)
- Iterators, [63](#)
  - \_\_iterator\_category, [65](#)
  - back\_inserter, [66](#)
  - front\_inserter, [66](#)
  - inserter, [66](#)
  - operator!=, [66](#)
  - operator==, [67](#)
- iterators\_fn\_imps.hpp, [1957](#), [1958](#)
- iword
  - std::basic\_ios, [1114](#)
  - std::ios\_base, [1412](#)
- k
  - std::negative\_binomial\_distribution, [1581](#)
- key\_comp
  - std::map, [1480](#)
  - std::multimap, [1556](#)
  - std::multiset, [1574](#)
  - std::set, [1700](#)
- key\_compare
  - std::set, [1685](#)
- key\_eq
  - std::unordered\_map, [1775](#)
  - std::unordered\_multimap, [1796](#)
  - std::unordered\_multiset, [1818](#)
  - std::unordered\_set, [1838](#)
- key\_equal
  - std::unordered\_map, [1762](#)
  - std::unordered\_multimap, [1782](#)
  - std::unordered\_multiset, [1803](#)
  - std::unordered\_set, [1823](#)
- key\_type
  - std::set, [1685](#)
  - std::unordered\_map, [1762](#)
  - std::unordered\_multimap, [1782](#)
  - std::unordered\_multiset, [1804](#)

- std::unordered\_set, 1823
- kill\_dependency
  - Atomics, 17
- L1\_cache\_size
  - \_\_gnu\_parallel::Settings, 671
- L2\_cache\_size
  - \_\_gnu\_parallel::Settings, 671
- lambda
  - std::exponential\_distribution, 1336
- left
  - std, 375
  - std::basic\_ios, 1122
  - std::ios\_base, 1417
- left\_child\_next\_sibling\_heap.hpp, 1958
- left\_child\_next\_sibling\_heap\_const\_iterator\_
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_↔  
const\_iterator\_, 782
- left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_↔
  - 
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_↔  
node\_point\_const\_iterator\_, 786
- length
  - \_\_gnu\_cxx::\_\_versa\_string, 491
  - std::basic\_string, 1162
  - std::match\_results, 1493
  - std::regex\_traits, 1673
  - std::sub\_match, 1724
- lexicographical\_compare
  - Sorting, 194
- linear\_congruential\_engine
  - std::linear\_congruential\_engine, 1430
- linear\_probe\_fn\_imp.hpp, 1958
- list
  - std::list, 1437, 1438
- List-Based, 68
- list\_partition
  - \_\_gnu\_parallel, 282
- list\_partition.h, 1959
- list\_update
  - \_\_gnu\_pbds::list\_update, 886
- list\_update\_policy.hpp, 1959
- load\_factor
  - std::unordered\_map, 1775
  - std::unordered\_multimap, 1796
  - std::unordered\_multiset, 1818
  - std::unordered\_set, 1838
- local\_iterator
  - std::unordered\_map, 1762
  - std::unordered\_multimap, 1782
  - std::unordered\_multiset, 1804
  - std::unordered\_set, 1823
- locale
  - std::locale, 1453, 1454
- locale\_classes.h, 1959
- locale\_facets.h, 1960
- locale\_facets\_nonio.h, 1961
- localefwd.h, 1961
- Locales, 69
- lookup\_classname
  - std::regex\_traits, 1673
- lookup\_collatename
  - std::regex\_traits, 1674
- losertree.h, 1962
- lower\_bound
  - Binary Search, 32, 33
  - std::map, 1481
  - std::multimap, 1556
  - std::multiset, 1574
  - std::set, 1700
- lu\_counter\_metadata.hpp, 1963
- lu\_map.hpp, 1963
- m\_p\_tbl
  - const\_iterator\_, 955
  - iterator\_, 959
- macros.h, 1964
  - \_GLIBCXX\_DEBUG\_VERIFY\_AT, 1966
  - \_\_glibcxx\_check\_erase, 1965
  - \_\_glibcxx\_check\_erase\_after, 1965
  - \_\_glibcxx\_check\_erase\_range, 1965
  - \_\_glibcxx\_check\_erase\_range\_after, 1965
  - \_\_glibcxx\_check\_heap\_pred, 1965
  - \_\_glibcxx\_check\_insert, 1965
  - \_\_glibcxx\_check\_insert\_after, 1965
  - \_\_glibcxx\_check\_insert\_range, 1965
  - \_\_glibcxx\_check\_insert\_range\_after, 1966
  - \_\_glibcxx\_check\_partitioned\_lower, 1966
  - \_\_glibcxx\_check\_partitioned\_lower\_pred, 1966
  - \_\_glibcxx\_check\_partitioned\_upper\_pred, 1966
  - \_\_glibcxx\_check\_sorted\_pred, 1966
- make\_exception\_ptr
  - Exceptions, 47
- make\_heap
  - Heap, 55
- make\_pair
  - Utilities, 220
- make\_shared
  - Pointer abstractions, 125
- malloc\_allocator.h, 1966
- map
  - std::map, 1470, 1471
- map.h, 1967, 1968
- mapped\_type
  - std::unordered\_map, 1762
  - std::unordered\_multimap, 1782
- mark\_count
  - std::basic\_regex, 1130

- mask\_array
  - Numeric\_arrays, 115
- mask\_array.h, 1968
- mask\_based\_range\_hashing.hpp, 1969
- match\_any
  - Regular Expressions, 177
- match\_continuous
  - Regular Expressions, 177
- match\_default
  - Regular Expressions, 177
- match\_flag\_type
  - Regular Expressions, 145
- match\_not\_bool
  - Regular Expressions, 177
- match\_not\_bow
  - Regular Expressions, 177
- match\_not\_eol
  - Regular Expressions, 177
- match\_not\_eow
  - Regular Expressions, 177
- match\_not\_null
  - Regular Expressions, 177
- match\_prev\_avail
  - Regular Expressions, 177
- match\_results
  - std::match\_results, 1491
- max
  - \_\_gnu\_parallel, 283
  - Sorting, 194, 196
  - std::bernoulli\_distribution, 1178
  - std::binomial\_distribution, 1187
  - std::cauchy\_distribution, 1191
  - std::chi\_squared\_distribution, 1198
  - std::discard\_block\_engine, 1325
  - std::discrete\_distribution, 1329
  - std::exponential\_distribution, 1336
  - std::extreme\_value\_distribution, 1339
  - std::fisher\_f\_distribution, 1342
  - std::gamma\_distribution, 1369
  - std::geometric\_distribution, 1372
  - std::independent\_bits\_engine, 1399
  - std::linear\_congruential\_engine, 1431
  - std::lognormal\_distribution, 1465
  - std::mersenne\_twister\_engine, 1503
  - std::negative\_binomial\_distribution, 1581
  - std::normal\_distribution, 1585
  - std::piecewise\_constant\_distribution, 1639
  - std::piecewise\_linear\_distribution, 1643
  - std::poisson\_distribution, 1652
  - std::shuffle\_order\_engine, 1712
  - std::student\_t\_distribution, 1719
  - std::uniform\_int\_distribution, 1751
  - std::uniform\_real\_distribution, 1754
  - std::weibull\_distribution, 1864
- max\_bucket\_count
  - std::unordered\_map, 1775
  - std::unordered\_multimap, 1796
  - std::unordered\_multiset, 1818
  - std::unordered\_set, 1838
- max\_count
  - \_\_gnu\_pbds::lu\_counter\_policy, 889
- max\_element
  - Sorting, 196
- max\_element\_minimal\_n
  - \_\_gnu\_parallel::Settings, 671
- max\_load\_factor
  - std::unordered\_map, 1776
  - std::unordered\_multimap, 1796
  - std::unordered\_multiset, 1818
  - std::unordered\_set, 1838
- max\_size
  - \_\_gnu\_cxx::\_\_alloc\_traits, 437
  - \_\_gnu\_cxx::\_\_versa\_string, 491
  - std::\_\_detail::\_\_Nfa, 1042
  - std::allocator\_traits, 1092
  - std::basic\_string, 1162
  - std::deque, 1318
  - std::forward\_list, 1358
  - std::list, 1445
  - std::map, 1481
  - std::match\_results, 1493
  - std::multimap, 1556
  - std::multiset, 1574
  - std::set, 1700
  - std::unordered\_map, 1776
  - std::unordered\_multimap, 1798
  - std::unordered\_multiset, 1818
  - std::unordered\_set, 1838
  - std::vector, 1854
- mean
  - std::normal\_distribution, 1585
  - std::poisson\_distribution, 1652
- memory\_order
  - Atomics, 17
- memoryfwd.h, 1969
- merge
  - Sorting, 198
  - std::forward\_list, 1358
  - std::list, 1445
- merge.h, 1969
- merge\_minimal\_n
  - \_\_gnu\_parallel::Settings, 671
- merge\_oversampling
  - \_\_gnu\_parallel::Settings, 671
- mersenne\_twister\_engine
  - std::mersenne\_twister\_engine, 1503
- messages
  - std::locale, 1457

- std::messages, 1507
- messages\_members.h, 1970
- metadata\_const\_reference
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_↔  
it\_, 725
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_, 729
- metadata\_reference
  - \_\_gnu\_pbds::lu\_counter\_policy, 889
  - \_\_gnu\_pbds::lu\_move\_to\_front\_policy, 890
- metadata\_type
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_↔  
it\_, 725
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_, 729
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer, 817
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_iter, 820
  - \_\_gnu\_pbds::lu\_counter\_policy, 889
  - \_\_gnu\_pbds::lu\_move\_to\_front\_policy, 890
  - \_\_gnu\_pbds::sample\_update\_policy, 913
- min
  - \_\_gnu\_parallel, 283
  - Sorting, 199
  - std::bernoulli\_distribution, 1178
  - std::binomial\_distribution, 1187
  - std::cauchy\_distribution, 1191
  - std::chi\_squared\_distribution, 1198
  - std::discard\_block\_engine, 1326
  - std::discrete\_distribution, 1329
  - std::exponential\_distribution, 1336
  - std::extreme\_value\_distribution, 1339
  - std::fisher\_f\_distribution, 1342
  - std::gamma\_distribution, 1369
  - std::geometric\_distribution, 1372
  - std::independent\_bits\_engine, 1399
  - std::linear\_congruential\_engine, 1431
  - std::lognormal\_distribution, 1465
  - std::mersenne\_twister\_engine, 1503
  - std::negative\_binomial\_distribution, 1581
  - std::normal\_distribution, 1585
  - std::piecewise\_constant\_distribution, 1639
  - std::piecewise\_linear\_distribution, 1643
  - std::poisson\_distribution, 1652
  - std::shuffle\_order\_engine, 1712
  - std::student\_t\_distribution, 1719
  - std::uniform\_int\_distribution, 1752
  - std::uniform\_real\_distribution, 1755
  - std::weibull\_distribution, 1864
- min\_element
  - Sorting, 199, 200
- min\_element\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 671
- minmax
  - Sorting, 200
- minmax\_element
  - Sorting, 200, 201
- minstd\_rand
  - Random Number Generators, 135
- minstd\_rand0
  - Random Number Generators, 135
- mismatch
  - Non-Mutating, 103
- mod\_based\_range\_hashing.hpp, 1970
- modulus
  - std::linear\_congruential\_engine, 1433
- monetary
  - std::locale, 1457
- money\_get
  - std::money\_get, 1516
- money\_put
  - std::money\_put, 1520
- money\_punct
  - std::money\_punct, 1525, 1526
- move
  - Mutating, 77
  - Utilities, 220
- move.h, 1970
- move\_backward
  - Mutating, 77
- move\_if\_noexcept
  - Utilities, 220
- mt19937
  - Random Number Generators, 135
- mt19937\_64
  - Random Number Generators, 135
- mt\_allocator.h, 1971
- multimap
  - std::multimap, 1544, 1546
- multimap.h, 1972, 1973
- multiplier
  - std::linear\_congruential\_engine, 1433
- multiseq\_partition
  - \_\_gnu\_parallel, 283
- multiseq\_selection
  - \_\_gnu\_parallel, 283
- multiseq\_selection.h, 1973
- multiset
  - std::multiset, 1562–1564
- multiset.h, 1974, 1975
- multiway\_merge
  - \_\_gnu\_parallel, 284
- multiway\_merge.h, 1975
- \_\_GLIBCXX\_\_PARALLEL\_LENGTH, 1978
- multiway\_merge\_3\_variant
  - \_\_gnu\_parallel, 285
- multiway\_merge\_4\_variant
  - \_\_gnu\_parallel, 286
- multiway\_merge\_exact\_splitting
  - \_\_gnu\_parallel, 286
- multiway\_merge\_loser\_tree

- [\\_\\_gnu\\_parallel](#), 287
- [multiway\\_merge\\_loser\\_tree\\_sentinel](#)
  - [\\_\\_gnu\\_parallel](#), 287
- [multiway\\_merge\\_loser\\_tree\\_unguarded](#)
  - [\\_\\_gnu\\_parallel](#), 288
- [multiway\\_merge\\_minimal\\_k](#)
  - [\\_\\_gnu\\_parallel::Settings](#), 671
- [multiway\\_merge\\_minimal\\_n](#)
  - [\\_\\_gnu\\_parallel::Settings](#), 671
- [multiway\\_merge\\_oversampling](#)
  - [\\_\\_gnu\\_parallel::Settings](#), 671
- [multiway\\_merge\\_sampling\\_splitting](#)
  - [\\_\\_gnu\\_parallel](#), 288
- [multiway\\_merge\\_sentinels](#)
  - [\\_\\_gnu\\_parallel](#), 288
- [multiway\\_mergesort.h](#), 1978
- [Mutating](#), 70
  - [copy](#), 72
  - [copy\\_backward](#), 72
  - [copy\\_if](#), 72
  - [copy\\_n](#), 73
  - [fill](#), 73
  - [fill\\_n](#), 73
  - [generate](#), 75
  - [generate\\_n](#), 75
  - [is\\_partitioned](#), 75
  - [iter\\_swap](#), 77
  - [move](#), 77
  - [move\\_backward](#), 77
  - [partition](#), 79
  - [partition\\_copy](#), 79
  - [partition\\_point](#), 80
  - [random\\_shuffle](#), 80
  - [remove](#), 81
  - [remove\\_copy](#), 81
  - [remove\\_copy\\_if](#), 81
  - [remove\\_if](#), 82
  - [replace](#), 82
  - [replace\\_copy\\_if](#), 83
  - [replace\\_if](#), 83
  - [reverse](#), 83
  - [reverse\\_copy](#), 85
  - [rotate](#), 85
  - [rotate\\_copy](#), 86
  - [shuffle](#), 86
  - [stable\\_partition](#), 86
  - [swap\\_ranges](#), 88
  - [transform](#), 88, 89
  - [unique](#), 89
  - [unique\\_copy](#), 91
- [name](#)
  - [std::locale](#), 1455
- [narrow](#)
  - [std::\\_\\_ctype\\_abstract\\_base](#), 979
  - [std::basic\\_ios](#), 1114
  - [std::ctype](#), 1244
  - [std::ctype< char >](#), 1256
  - [std::ctype< wchar\\_t >](#), 1270, 1271
  - [std::ctype\\_byname](#), 1285
  - [std::ctype\\_byname< char >](#), 1295, 1296
- [neg\\_format](#)
  - [std::moneypunct](#), 1530
  - [std::moneypunct\\_byname](#), 1538
- [negative\\_sign](#)
  - [std::moneypunct](#), 1531
  - [std::moneypunct\\_byname](#), 1539
- [Negators](#), 93
  - [not1](#), 93
  - [not2](#), 93
- [nested\\_exception.h](#), 1979
- [new\\_allocator.h](#), 1979
- [next\\_permutation](#)
  - [Sorting](#), 201
- [noboolalpha](#)
  - [std](#), 375
- [node.hpp](#), 1980
- [node\\_begin](#)
  - [\\_\\_gnu\\_pbds::detail::ov\\_tree\\_map](#), 795
  - [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_map](#), 824
  - [\\_\\_gnu\\_pbds::detail::rb\\_tree\\_map](#), 834
  - [\\_\\_gnu\\_pbds::detail::splay\\_tree\\_map](#), 842
- [node\\_const\\_iterator](#)
  - [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_traits](#), 732
  - [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_traits< Key, null\\_type, Cmp\\_Fn, Node\\_Update, Node, \\_Alloc >](#), 732
  - [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, ov\\_tree\\_tag, \\_Alloc >](#), 852
  - [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, rb\\_tree\\_tag, \\_Alloc >](#), 854
  - [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, splay\\_tree\\_tag, \\_Alloc >](#), 855
  - [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, null\\_type, Cmp\\_Fn, Node\\_Update, ov\\_tree\\_tag, \\_Alloc >](#), 856
  - [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, null\\_type, Cmp\\_Fn, Node\\_Update, rb\\_tree\\_tag, \\_Alloc >](#), 857
  - [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, null\\_type, Cmp\\_Fn, Node\\_Update, splay\\_tree\\_tag, \\_Alloc >](#), 859
  - [\\_\\_gnu\\_pbds::detail::trie\\_traits< Key, Mapped, \\_ATraits, Node\\_Update, pat\\_trie\\_tag, \\_Alloc >](#), 863

- \_\_gnu\_pbds::detail::trie\_traits< Key, null\_type, \_↔  
ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >, 864
- node\_end
  - \_\_gnu\_pbds::detail::ov\_tree\_map, 795
  - \_\_gnu\_pbds::detail::pat\_trie\_map, 824
  - \_\_gnu\_pbds::detail::rb\_tree\_map, 834
  - \_\_gnu\_pbds::detail::splay\_tree\_map, 842
- node\_iterators.hpp, 1981
- node\_metadata\_selector.hpp, 1982
- node\_type
  - \_\_gnu\_pbds::detail::pat\_trie\_base, 803
  - \_\_gnu\_pbds::detail::pat\_trie\_map, 824
- node\_update
  - \_\_gnu\_pbds::detail::trie\_traits< Key, Mapped, \_↔  
ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >, 863
  - \_\_gnu\_pbds::detail::trie\_traits< Key, null\_type, \_↔  
ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >, 864
- Non-Mutating, 95
  - adjacent\_find, 96
  - all\_of, 97
  - any\_of, 97
  - count, 97
  - count\_if, 98
  - equal, 98
  - find, 99
  - find\_end, 99, 100
  - find\_first\_of, 100, 101
  - find\_if, 101
  - find\_if\_not, 101
  - for\_each, 102
  - is\_permutation, 102
  - mismatch, 103
  - none\_of, 103
  - search, 105
  - search\_n, 106
- none
  - std::locale, 1457
- none\_of
  - Non-Mutating, 103
- Normal Distributions, 108
  - operator!=, 109
  - operator<<, 110
  - operator>>, 110
- normal\_distribution
  - std::normal\_distribution, 1585
- noshowbase
  - std, 375
- noshowpoint
  - std, 376
- noshowpos
  - std, 376
- noskipws
  - std, 376
- nosubs
  - Regular Expressions, 178
- not1
  - Negators, 93
- not2
  - Negators, 93
- notify\_cleared
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_↔  
resize\_trigger, 707
  - \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger, 880
  - \_\_gnu\_pbds::sample\_resize\_policy, 906
  - \_\_gnu\_pbds::sample\_resize\_trigger, 908
- notify\_erase\_search\_collision
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_↔  
resize\_trigger, 707
  - \_\_gnu\_pbds::sample\_resize\_policy, 906
  - \_\_gnu\_pbds::sample\_resize\_trigger, 908
- notify\_erase\_search\_end
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_↔  
resize\_trigger, 707
  - \_\_gnu\_pbds::sample\_resize\_policy, 906
  - \_\_gnu\_pbds::sample\_resize\_trigger, 908
- notify\_erase\_search\_start
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_↔  
resize\_trigger, 707
  - \_\_gnu\_pbds::sample\_resize\_policy, 906
  - \_\_gnu\_pbds::sample\_resize\_trigger, 908
- notify\_erased
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_↔  
resize\_trigger, 707
  - \_\_gnu\_pbds::sample\_resize\_policy, 906
  - \_\_gnu\_pbds::sample\_resize\_trigger, 908
- notify\_externally\_resized
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_↔  
resize\_trigger, 708
  - \_\_gnu\_pbds::sample\_resize\_trigger, 908
- notify\_find\_search\_collision
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_↔  
resize\_trigger, 708
  - \_\_gnu\_pbds::sample\_resize\_policy, 906
  - \_\_gnu\_pbds::sample\_resize\_trigger, 908
- notify\_find\_search\_end
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_↔  
resize\_trigger, 708
  - \_\_gnu\_pbds::sample\_resize\_policy, 906
  - \_\_gnu\_pbds::sample\_resize\_trigger, 909
- notify\_find\_search\_start
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_↔  
resize\_trigger, 708
  - \_\_gnu\_pbds::sample\_resize\_policy, 906
  - \_\_gnu\_pbds::sample\_resize\_trigger, 909
- notify\_insert\_search\_collision



- \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_↔  
resize\_trigger, 708
  - \_\_gnu\_pbds::sample\_resize\_policy, 906
  - \_\_gnu\_pbds::sample\_resize\_trigger, 909
- notify\_insert\_search\_end
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_↔  
resize\_trigger, 708
  - \_\_gnu\_pbds::sample\_resize\_policy, 906
  - \_\_gnu\_pbds::sample\_resize\_trigger, 909
- notify\_insert\_search\_start
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_↔  
resize\_trigger, 708
  - \_\_gnu\_pbds::sample\_resize\_policy, 906
  - \_\_gnu\_pbds::sample\_resize\_trigger, 909
- notify\_inserted
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_↔  
resize\_trigger, 709
  - \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger, 880
  - \_\_gnu\_pbds::sample\_resize\_policy, 907
  - \_\_gnu\_pbds::sample\_resize\_trigger, 909
- notify\_resized
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_↔  
resize\_trigger, 709
  - \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger, 880
  - \_\_gnu\_pbds::sample\_range\_hashing, 903
  - \_\_gnu\_pbds::sample\_ranged\_hash\_fn, 904
  - \_\_gnu\_pbds::sample\_resize\_policy, 907
  - \_\_gnu\_pbds::sample\_resize\_trigger, 909
- nounitbuf
  - std, 376
- nouppercase
  - std, 376
- npos
  - \_\_gnu\_cxx::\_\_versa\_string, 508
  - std::basic\_string, 1176
- nth\_element
  - Sorting, 203
- nth\_element\_minimal\_n
  - \_\_gnu\_parallel::Settings, 671
- null\_node\_metadata.hpp, 1982
- num\_children
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer, 817
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_iter, 820
- num\_get
  - std::num\_get, 1591
- num\_put
  - std::num\_put, 1605
- numeric
  - std::locale, 1457
- Numeric\_arrays, 111
  - ~gslice, 115
  - gslice, 114
  - gslice\_array, 114
  - indirect\_array, 115
  - mask\_array, 115
  - operator<=, 118
  - operator>=, 120
  - operator\*=, 116, 117
  - operator^=, 120, 121
  - operator+=, 117
  - operator-=, 117, 118
  - operator/=, 118
  - operator=, 119, 120
  - operator%=, 115, 116
  - operator&=, 116
  - operator|=, 121
  - size, 121
  - slice, 115
  - slice\_array, 115
  - start, 121, 122
  - stride, 122
- numeric\_traits.h, 1983
- numeric\_fwd.h, 1983
- numpunct
  - std::numpunct, 1617, 1618
- oct
  - std, 376
  - std::basic\_ios, 1122
  - std::ios\_base, 1417
- off\_type
  - std::basic\_ios, 1108
- omp\_loop.h, 1985
- omp\_loop\_static.h, 1985
- openmode
  - std::basic\_ios, 1108
  - std::ios\_base, 1409
- operator\_iterator
  - \_\_gnu\_debug::\_Safe\_iterator, 554
  - \_\_gnu\_debug::\_Safe\_local\_iterator, 565
- operator\_RAlter
  - \_\_gnu\_parallel::\_GuardedIterator, 626
- operator\_const\_point\_iterator\_
  - iterator\_, 958
- operator\_point\_iterator\_
  - iterator\_, 958
- operator\_streamoff
  - std::fpos, 1364
- operator\_string\_type
  - std::sub\_match, 1724
- operator\_void\_\*
  - std::basic\_ios, 1114
- operator!
  - std::basic\_ios, 1115
- operator!=
  - \_\_gnu\_cxx, 230
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_↔  
it\_, 726



- \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it, 730
- \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_, 737
- \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_, 741
- \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator\_, 782
- \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_, 786
- \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer, 817
- \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_iter, 820
- Bernoulli Distributions, 28
- const\_iterator\_, 954
- iterator\_, 958
- Iterators, 66
- Normal Distributions, 109
- point\_const\_iterator\_, 961
- point\_iterator\_, 964
- Poisson Distributions, 128, 129
- Random Number Generators, 136, 137
- Regular Expressions, 148, 149, 151
- std, 376–378
- std::locale, 1455
- std::regex\_iterator, 1666
- std::regex\_token\_iterator, 1670
- std::rel\_ops, 430
- Uniform Distributions, 214
- Utilities, 221
- operator<
  - \_\_gnu\_cxx, 233, 234
  - \_\_gnu\_parallel::\_GuardedIterator, 627
  - Regular Expressions, 151, 153, 154
  - std, 382–384, 386, 387
  - Utilities, 221
- operator<<
  - Bernoulli Distributions, 28
  - Normal Distributions, 110
  - Pointer\_abstractions, 126
  - Poisson Distributions, 129
  - Random Number Generators, 138
  - Regular Expressions, 154
  - std, 387
  - std::binomial\_distribution, 1189
  - std::chi\_squared\_distribution, 1199
  - std::discard\_block\_engine, 1326
  - std::discrete\_distribution, 1329
  - std::fisher\_f\_distribution, 1344
  - std::gamma\_distribution, 1370
  - std::linear\_congruential\_engine, 1432
  - std::lognormal\_distribution, 1465
  - std::mersenne\_twister\_engine, 1504
  - std::negative\_binomial\_distribution, 1582
  - std::normal\_distribution, 1586
  - std::piecewise\_constant\_distribution, 1640
- std::piecewise\_linear\_distribution, 1644
- std::poisson\_distribution, 1653
- std::shuffle\_order\_engine, 1713
- std::student\_t\_distribution, 1720
- Uniform Distributions, 214, 216
- operator<<=
  - Numeric\_arrays, 118
- operator<=
  - \_\_gnu\_cxx, 234, 235
  - \_\_gnu\_parallel::\_GuardedIterator, 628
  - Regular Expressions, 154–156
  - std, 388–390
  - std::rel\_ops, 431
  - Utilities, 221
- operator>
  - \_\_gnu\_cxx, 236, 237
  - Regular Expressions, 160, 161, 163
  - std, 396, 397, 399
  - std::rel\_ops, 431
  - Utilities, 221
- operator>>
  - Bernoulli Distributions, 29
  - Normal Distributions, 110
  - Poisson Distributions, 130
  - std, 402, 403
  - std::binomial\_distribution, 1189
  - std::chi\_squared\_distribution, 1199
  - std::discard\_block\_engine, 1327
  - std::discrete\_distribution, 1331
  - std::fisher\_f\_distribution, 1344
  - std::gamma\_distribution, 1370
  - std::independent\_bits\_engine, 1400
  - std::linear\_congruential\_engine, 1432
  - std::lognormal\_distribution, 1466
  - std::mersenne\_twister\_engine, 1504
  - std::negative\_binomial\_distribution, 1582
  - std::normal\_distribution, 1586
  - std::piecewise\_constant\_distribution, 1640
  - std::piecewise\_linear\_distribution, 1644
  - std::poisson\_distribution, 1654
  - std::shuffle\_order\_engine, 1714
  - std::student\_t\_distribution, 1721
  - Uniform Distributions, 216
- operator>>=
  - Numeric\_arrays, 120
- operator>=
  - \_\_gnu\_cxx, 237, 238
  - Regular Expressions, 163, 164, 166
  - std, 399, 400, 402
  - std::rel\_ops, 431
  - Utilities, 221
- operator\*
  - \_\_gnu\_debug::\_Safe\_iterator, 554
  - \_\_gnu\_debug::\_Safe\_local\_iterator, 565

- `__gnu_parallel::__GuardedIterator`, 626
- `__gnu_pbds::detail::bin_search_tree_const_node_↵  
it_`, 726
- `__gnu_pbds::detail::bin_search_tree_node_it_`, 730
- `__gnu_pbds::detail::binary_heap_const_iterator_`, 737
- `__gnu_pbds::detail::binary_heap_point_const_↵  
iterator_`, 741
- `__gnu_pbds::detail::left_child_next_sibling_heap_↵  
const_iterator_`, 782
- `__gnu_pbds::detail::left_child_next_sibling_heap_↵  
node_point_const_iterator_`, 786
- `__gnu_pbds::detail::ov_tree_node_it_`, 799
- `__gnu_pbds::detail::pat_trie_base::_Node_citer`, 818
- `__gnu_pbds::detail::pat_trie_base::_Node_iter`, 821
- `const_iterator_`, 954
- `iterator_`, 958
- `point_const_iterator_`, 961
- `point_iterator_`, 964
- `std::auto_ptr`, 1098
- `std::back_insert_iterator`, 1101
- `std::front_insert_iterator`, 1367
- `std::insert_iterator`, 1404
- `std::istreambuf_iterator`, 1424
- `std::ostreambuf_iterator`, 1632
- `std::regex_iterator`, 1666
- `std::regex_token_iterator`, 1670
- `std::reverse_iterator`, 1679
- `operator*=`
  - `Numeric_arrays`, 116, 117
- `operator^=`
  - `Numeric_arrays`, 120, 121
- `operator()`
  - `__gnu_parallel::__Nothing`, 657
  - `__gnu_parallel::__RandomNumber`, 665
  - `__gnu_parallel::__accumulate_selector`, 585
  - `__gnu_parallel::__adjacent_find_selector`, 588
  - `__gnu_parallel::__count_if_selector`, 592
  - `__gnu_parallel::__count_selector`, 593
  - `__gnu_parallel::__fill_selector`, 594
  - `__gnu_parallel::__find_first_of_selector`, 596
  - `__gnu_parallel::__find_if_selector`, 597
  - `__gnu_parallel::__for_each_selector`, 598
  - `__gnu_parallel::__generate_selector`, 599
  - `__gnu_parallel::__identity_selector`, 604
  - `__gnu_parallel::__inner_product_selector`, 606
  - `__gnu_parallel::__mismatch_selector`, 608
  - `__gnu_parallel::__replace_if_selector`, 613
  - `__gnu_parallel::__replace_selector`, 615
  - `__gnu_parallel::__transform1_selector`, 616
  - `__gnu_parallel::__transform2_selector`, 618
  - `__gnu_pbds::direct_mask_range_hashing`, 870
  - `__gnu_pbds::direct_mod_range_hashing`, 872
  - `__gnu_pbds::linear_probe_fn`, 886
  - `__gnu_pbds::lu_counter_policy`, 889
  - `__gnu_pbds::lu_move_to_front_policy`, 890
  - `__gnu_pbds::quadratic_probe_fn`, 898
  - `__gnu_pbds::sample_probe_fn`, 902
  - `__gnu_pbds::sample_range_hashing`, 903
  - `__gnu_pbds::sample_ranged_hash_fn`, 904
  - `__gnu_pbds::sample_trie_node_update`, 912
  - `__gnu_pbds::sample_update_policy`, 913
  - `__gnu_pbds::tree_order_statistics_node_update`, 921
  - `__gnu_pbds::trie_order_statistics_node_update`, 927
  - `__gnu_pbds::trie_prefix_search_node_update`, 930
  - `std::bernoulli_distribution`, 1178
  - `std::binomial_distribution`, 1187
  - `std::cauchy_distribution`, 1192
  - `std::chi_squared_distribution`, 1198
  - `std::discard_block_engine`, 1326
  - `std::discrete_distribution`, 1329
  - `std::exponential_distribution`, 1336
  - `std::extreme_value_distribution`, 1339
  - `std::fisher_f_distribution`, 1342
  - `std::gamma_distribution`, 1369
  - `std::geometric_distribution`, 1373
  - `std::independent_bits_engine`, 1399
  - `std::linear_congruential_engine`, 1431
  - `std::locale`, 1455
  - `std::lognormal_distribution`, 1465
  - `std::negative_binomial_distribution`, 1581
  - `std::normal_distribution`, 1585
  - `std::piecewise_constant_distribution`, 1639
  - `std::piecewise_linear_distribution`, 1643
  - `std::poisson_distribution`, 1653
  - `std::shuffle_order_engine`, 1713
  - `std::student_t_distribution`, 1720
  - `std::uniform_int_distribution`, 1752
  - `std::uniform_real_distribution`, 1755
  - `std::weibull_distribution`, 1864
- `operator+`
  - `__gnu_cxx`, 231, 233
  - `std`, 378, 380
  - `std::fpos`, 1364
  - `std::reverse_iterator`, 1680
- `operator++`
  - `__gnu_debug::_Safe_iterator`, 554
  - `__gnu_debug::_Safe_local_iterator`, 565
  - `__gnu_parallel::__GuardedIterator`, 626
  - `const_iterator_`, 954
  - `iterator_`, 958
  - `std::back_insert_iterator`, 1102
  - `std::front_insert_iterator`, 1367
  - `std::insert_iterator`, 1405
  - `std::istreambuf_iterator`, 1424
  - `std::ostreambuf_iterator`, 1632
  - `std::regex_iterator`, 1666

- std::regex\_token\_iterator, 1671
- std::reverse\_iterator, 1680
- operator+=
  - \_\_gnu\_cxx::\_\_versa\_string, 491, 493
  - Numeric\_arrays, 117
  - std::basic\_string, 1162, 1164
  - std::fpos, 1364
  - std::reverse\_iterator, 1680
- operator-
  - std::fpos, 1364
  - std::reverse\_iterator, 1680
- operator->
  - \_\_gnu\_debug::\_\_Safe\_iterator, 555
  - \_\_gnu\_debug::\_\_Safe\_local\_iterator, 566
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_, 737
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_, 741
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator\_, 782
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_, 786
  - const\_iterator\_, 954
  - iterator\_, 958
  - point\_const\_iterator\_, 961
  - point\_iterator\_, 964
  - std::auto\_ptr, 1098
  - std::regex\_iterator, 1666
  - std::regex\_token\_iterator, 1671
  - std::reverse\_iterator, 1681
- operator--
  - \_\_gnu\_debug::\_\_Safe\_iterator, 554
  - std::reverse\_iterator, 1681
- operator-=
  - Numeric\_arrays, 117, 118
  - std::fpos, 1364
  - std::reverse\_iterator, 1681
- operator/=
  - Numeric\_arrays, 118
- operator=
  - \_\_gnu\_cxx::\_\_versa\_string, 493, 495
  - \_\_gnu\_debug::\_\_Safe\_iterator, 555
  - \_\_gnu\_debug::\_\_Safe\_local\_iterator, 566
  - Numeric\_arrays, 119, 120
  - std::auto\_ptr, 1098
  - std::back\_insert\_iterator, 1102
  - std::basic\_regex, 1130
  - std::basic\_string, 1164, 1165
  - std::deque, 1318, 1320
  - std::forward\_list, 1358, 1359
  - std::front\_insert\_iterator, 1367
  - std::insert\_iterator, 1405
  - std::list, 1446
  - std::locale, 1456
  - std::map, 1481, 1482
  - std::match\_results, 1493, 1494
  - std::multimap, 1557
  - std::multiset, 1574, 1576
  - std::ostream\_iterator, 1629
  - std::ostreambuf\_iterator, 1632
  - std::regex\_iterator, 1666
  - std::regex\_token\_iterator, 1671
  - std::set, 1700, 1701
  - std::unordered\_map, 1776
  - std::unordered\_multimap, 1798
  - std::unordered\_multiset, 1818, 1819
  - std::unordered\_set, 1838, 1839
  - std::vector, 1854
- operator==
  - \_\_gnu\_cxx, 235, 236
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_iterator\_, 726
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_iterator\_, 731
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_, 738
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_, 741
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator\_, 783
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_, 787
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer, 818
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_iter, 821
  - const\_iterator\_, 955
  - iterator\_, 958
  - Iterators, 67
  - point\_const\_iterator\_, 962
  - point\_iterator\_, 964
  - Regular Expressions, 156, 158–160
  - std, 391–394, 396
  - std::bernoulli\_distribution, 1178
  - std::binomial\_distribution, 1189
  - std::cauchy\_distribution, 1192
  - std::chi\_squared\_distribution, 1199
  - std::discard\_block\_engine, 1327
  - std::discrete\_distribution, 1331
  - std::exponential\_distribution, 1337
  - std::extreme\_value\_distribution, 1340
  - std::fisher\_f\_distribution, 1344
  - std::gamma\_distribution, 1370
  - std::geometric\_distribution, 1373
  - std::independent\_bits\_engine, 1400
  - std::linear\_congruential\_engine, 1432
  - std::locale, 1456
  - std::lognormal\_distribution, 1466
  - std::mersenne\_twister\_engine, 1504
  - std::negative\_binomial\_distribution, 1582
  - std::normal\_distribution, 1586

- std::piecewise\_constant\_distribution, 1640
- std::piecewise\_linear\_distribution, 1644
- std::poisson\_distribution, 1654
- std::regex\_iterator, 1667
- std::regex\_token\_iterator, 1671
- std::shuffle\_order\_engine, 1713
- std::student\_t\_distribution, 1720
- std::uniform\_int\_distribution, 1752
- std::uniform\_real\_distribution, 1755
- std::weibull\_distribution, 1865
- Utilities, 221
- operator%=
  - Numeric\_arrays, 115, 116
- operator&=
  - Numeric\_arrays, 116
- operator[]
  - \_\_gnu\_cxx::\_\_versa\_string, 495
  - std::\_\_detail::\_\_Nfa, 1042, 1043
  - std::basic\_string, 1165, 1166
  - std::deque, 1320
  - std::map, 1482
  - std::match\_results, 1494
  - std::reverse\_iterator, 1681
  - std::unordered\_map, 1777
  - std::vector, 1856
- operator| =
  - Numeric\_arrays, 121
- opt\_random.h, 1985
- optimize
  - Regular Expressions, 178
- order\_of\_key
  - \_\_gnu\_pbds::tree\_order\_statistics\_node\_update, 921
  - \_\_gnu\_pbds::trie\_order\_statistics\_node\_update, 927
- order\_of\_prefix
  - \_\_gnu\_pbds::trie\_order\_statistics\_node\_update, 927
- order\_preserving
  - Traits, 213
- order\_statistics\_imp.hpp, 1986
- os\_defines.h, 1986
- ostream\_insert.h, 1986
- ostream\_iterator
  - std::ostream\_iterator, 1629
- ostream\_type
  - std::ostream\_iterator, 1628
  - std::ostreambuf\_iterator, 1631
- ostreambuf\_iterator
  - std::ostreambuf\_iterator, 1632
- out
  - std::\_\_codecvt\_abstract\_base, 970
  - std::basic\_ios, 1122
  - std::codecvt, 1203
  - std::codecvt< \_InternT, \_ExternT, encoding\_state >, 1207
  - std::codecvt< char, char, mbstate\_t >, 1211
  - std::codecvt< wchar\_t, char, mbstate\_t >, 1215
  - std::codecvt\_byname, 1220
  - std::ios\_base, 1417
- ov\_tree\_map\_.hpp, 1986
- p
  - std::bernoulli\_distribution, 1178
  - std::binomial\_distribution, 1187
  - std::geometric\_distribution, 1373
  - std::negative\_binomial\_distribution, 1581
- pair
  - std::pair, 1636, 1637
- pairing\_heap\_.hpp, 1987
- par\_loop.h, 1987
- parallel.h, 1988
- parallel\_balanced
  - \_\_gnu\_parallel, 254
- parallel\_multiway\_merge
  - \_\_gnu\_parallel, 290
- parallel\_omp\_loop
  - \_\_gnu\_parallel, 254
- parallel\_omp\_loop\_static
  - \_\_gnu\_parallel, 254
- parallel\_sort\_mwms
  - \_\_gnu\_parallel, 290
- parallel\_sort\_mwms\_pu
  - \_\_gnu\_parallel, 292
- parallel\_tag
  - \_\_gnu\_parallel::parallel\_tag, 693
- parallel\_taskqueue
  - \_\_gnu\_parallel, 254
- parallel\_unbalanced
  - \_\_gnu\_parallel, 254
- param
  - std::bernoulli\_distribution, 1178
  - std::binomial\_distribution, 1187
  - std::cauchy\_distribution, 1192
  - std::chi\_squared\_distribution, 1198
  - std::discrete\_distribution, 1329
  - std::exponential\_distribution, 1337
  - std::extreme\_value\_distribution, 1340
  - std::fisher\_f\_distribution, 1342
  - std::gamma\_distribution, 1369, 1370
  - std::geometric\_distribution, 1373
  - std::lognormal\_distribution, 1465
  - std::negative\_binomial\_distribution, 1581
  - std::normal\_distribution, 1585
  - std::piecewise\_constant\_distribution, 1639
  - std::piecewise\_linear\_distribution, 1643
  - std::poisson\_distribution, 1653
  - std::student\_t\_distribution, 1720
  - std::uniform\_int\_distribution, 1752
  - std::uniform\_real\_distribution, 1755

- std::weibull\_distribution, 1864
- partial\_sort
  - Sorting, 204
- partial\_sort\_copy
  - Sorting, 204, 206
- partial\_sort\_minimal\_n
  - \_\_gnu\_parallel::Settings, 672
- partial\_sum
  - std, 403
- partial\_sum.h, 1988
- partial\_sum\_dilation
  - \_\_gnu\_parallel::Settings, 672
- partial\_sum\_minimal\_n
  - \_\_gnu\_parallel::Settings, 672
- partition
  - Mutating, 79
- partition.h, 1988
  - \_GLIBCXX\_VOLATILE, 1989
- partition\_chunk\_share
  - \_\_gnu\_parallel::Settings, 672
- partition\_chunk\_size
  - \_\_gnu\_parallel::Settings, 672
- partition\_copy
  - Mutating, 79
- partition\_minimal\_n
  - \_\_gnu\_parallel::Settings, 672
- partition\_point
  - Mutating, 80
- pat\_trie.hpp, 1989
- pat\_trie\_base.hpp, 1990
- piecewise\_construct
  - Utilities, 222
- pod\_char\_traits.h, 1990
- point\_const\_iterator.hpp, 1991, 1992
- point\_const\_iterator\_, 959
  - const\_pointer, 960
  - const\_reference, 960
  - difference\_type, 960
  - iterator\_category, 960
  - operator!=, 961
  - operator\*, 961
  - operator->, 961
  - operator==, 962
  - point\_const\_iterator\_, 961
  - pointer, 960
  - reference, 961
  - value\_type, 961
- point\_iterator.hpp, 1992
- point\_iterator\_, 962
  - const\_pointer, 963
  - const\_reference, 963
  - difference\_type, 963
  - iterator\_category, 963
  - operator!=, 964

- operator\*, 964
- operator->, 964
- operator==, 964
- point\_iterator\_, 963
- pointer, 963
- reference, 963
- value\_type, 963
- point\_iterators.hpp, 1992
- pointer
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_, 736
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_, 740
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator\_, 781
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_, 785
  - const\_iterator\_, 953
  - iterator\_, 957
  - point\_const\_iterator\_, 960
  - point\_iterator\_, 963
  - std::allocator\_traits, 1090
  - std::back\_insert\_iterator, 1101
  - std::front\_insert\_iterator, 1366
  - std::insert\_iterator, 1404
  - std::istream\_iterator, 1420
  - std::istreambuf\_iterator, 1423
  - std::iterator, 1425
  - std::ostream\_iterator, 1628
  - std::ostreambuf\_iterator, 1631
  - std::pointer\_traits, 1650
  - std::pointer\_traits< \_Tp \* >, 1651
  - std::raw\_storage\_iterator, 1663
  - std::set, 1686
  - std::unordered\_map, 1762
  - std::unordered\_multimap, 1782
  - std::unordered\_multiset, 1804
  - std::unordered\_set, 1824
- pointer.h, 1993
- Pointer\_abstractions, 123
  - allocate\_shared, 125
  - get\_deleter, 125
  - make\_shared, 125
  - operator<<, 126
- pointer\_to
  - std::pointer\_traits< \_Tp \* >, 1651
- Poisson Distributions, 127
  - operator!=, 128, 129
  - operator<<, 129
  - operator>>, 130
- Policy-Based Data Structures, 131
- policy\_access\_fn\_imps.hpp, 1994, 1995
- pool\_allocator.h, 1995
- pop

- std::priority\_queue, 1657
- std::queue, 1660
- std::stack, 1717
- pop\_back
  - \_\_gnu\_cxx::\_\_versa\_string, 497
  - \_\_gnu\_parallel::\_\_RestrictedBoundedConcurrentQueue, 666
  - std::\_\_detail::\_\_Nfa, 1043
  - std::basic\_string, 1166
  - std::deque, 1320
  - std::list, 1446
  - std::vector, 1856
- pop\_front
  - \_\_gnu\_parallel::\_\_RestrictedBoundedConcurrentQueue, 666
  - std::deque, 1321
  - std::forward\_list, 1359
  - std::list, 1446
- pop\_heap
  - Heap, 55, 56
- pos\_format
  - std::moneypunct, 1531
  - std::moneypunct\_byname, 1539
- pos\_type
  - std::basic\_ios, 1108
- position
  - std::match\_results, 1494
- positive\_sign
  - std::moneypunct, 1531
  - std::moneypunct\_byname, 1540
- postypes.h, 1996
- precision
  - std::basic\_ios, 1115
  - std::ios\_base, 1412
- prefix
  - std::match\_results, 1494
- prefix\_range
  - \_\_gnu\_pbds::trie\_prefix\_search\_node\_update, 930
- prefix\_search\_node\_update\_imp.hpp, 1996
- prev\_permutation
  - Sorting, 206, 207
- priority\_queue
  - Heap-Based, 60
  - std::priority\_queue, 1656
- priority\_queue.hpp, 1996
- priority\_queue\_base\_dispatch.hpp, 1997
- probabilities
  - std::discrete\_distribution, 1329
- probe\_fn\_base.hpp, 1997
- profiler.h, 1997
- profiler\_algos.h, 2000
- profiler\_container\_size.h, 2000
- profiler\_hash\_func.h, 2001
- profiler\_hashtable\_size.h, 2001
- profiler\_list\_to\_slist.h, 2002
- profiler\_list\_to\_vector.h, 2002
- profiler\_map\_to\_unordered\_map.h, 2003
- profiler\_node.h, 2003
- profiler\_state.h, 2004
- profiler\_trace.h, 2004
- profiler\_vector\_size.h, 2006
- profiler\_vector\_to\_list.h, 2007
- propagate\_on\_container\_copy\_assignment
  - \_\_gnu\_cxx::\_\_alloc\_traits, 434
  - std::allocator\_traits, 1090
- propagate\_on\_container\_move\_assignment
  - \_\_gnu\_cxx::\_\_alloc\_traits, 434
  - std::allocator\_traits, 1090
- propagate\_on\_container\_swap
  - \_\_gnu\_cxx::\_\_alloc\_traits, 434
  - std::allocator\_traits, 1091
- ptr\_fun
  - Adaptors for pointers to functions, 5
- ptr\_traits.h, 2007
- push
  - std::priority\_queue, 1657
  - std::queue, 1660
  - std::stack, 1717
- push\_back
  - \_\_gnu\_cxx::\_\_versa\_string, 497
  - std::\_\_detail::\_\_Nfa, 1043
  - std::basic\_string, 1166
  - std::deque, 1321
  - std::list, 1447
  - std::vector, 1856
- push\_front
  - \_\_gnu\_parallel::\_\_RestrictedBoundedConcurrentQueue, 667
  - std::deque, 1321
  - std::forward\_list, 1359
  - std::list, 1447
- push\_heap
  - Heap, 56
- put
  - std::money\_put, 1521
  - std::num\_put, 1609–1614
  - std::time\_put, 1744
  - std::time\_put\_byname, 1747
- pword
  - std::basic\_ios, 1115
  - std::ios\_base, 1412
- qsb\_steals
  - \_\_gnu\_parallel::\_\_Settings, 672
- quadratic\_probe\_fn\_imp.hpp, 2008
- queue
  - std::queue, 1659
- queue.h, 2008

- `__GLIBCXX_VOLATILE`, 2008
- `quicksort.h`, 2008
- `r_erase_fn_imps.hpp`, 2009
- Random Number Distributions, 132
- Random Number Generation, 133
  - `generate_canonical`, 133
- Random Number Generators, 134
  - `minstd_rand`, 135
  - `minstd_rand0`, 135
  - `mt19937`, 135
  - `mt19937_64`, 135
  - `operator!=`, 136, 137
  - `operator<<`, 138
- Random Number Utilities, 139
- `random.h`, 2009
- `random_number.h`, 2013
- `random_shuffle`
  - Mutating, 80
- `random_shuffle.h`, 2013
- `random_shuffle_minimal_n`
  - `__gnu_parallel::Settings`, 672
- `range_access.h`, 2014
- `ranged_hash_fn.hpp`, 2014
- `ranged_probe_fn.hpp`, 2015
- `rb_tree_.hpp`, 2015
- `rbegin`
  - `__gnu_cxx::__versa_string`, 497
  - `std::__detail::__Nfa`, 1043, 1044
  - `std::basic_string`, 1166
  - `std::deque`, 1321
  - `std::list`, 1447
  - `std::map`, 1483
  - `std::multimap`, 1557
  - `std::multiset`, 1576
  - `std::set`, 1701
  - `std::vector`, 1857
- `rc.hpp`, 2016
- `rc_binomial_heap_.hpp`, 2016
- `rc_string_base.h`, 2016
- `rdbuf`
  - `std::basic_ios`, 1115, 1116
- `rdstate`
  - `std::basic_ios`, 1116
- `ready`
  - `std::match_results`, 1495
- reference
  - `__gnu_pbds::detail::bin_search_tree_const_node_↵`  
`it_`, 725
  - `__gnu_pbds::detail::bin_search_tree_node_it_`, 730
  - `__gnu_pbds::detail::binary_heap_const_iterator_`,  
736
  - `__gnu_pbds::detail::binary_heap_point_const_↵`  
`iterator_`, 740
  - `__gnu_pbds::detail::left_child_next_sibling_heap_↵`  
`const_iterator_`, 781
  - `__gnu_pbds::detail::left_child_next_sibling_heap_↵`  
`node_point_const_iterator_`, 785
  - `const_iterator_`, 953
  - `iterator_`, 957
  - `point_const_iterator_`, 961
  - `point_iterator_`, 963
  - `std::back_insert_iterator`, 1101
  - `std::front_insert_iterator`, 1366
  - `std::insert_iterator`, 1404
  - `std::istream_iterator`, 1420
  - `std::istreambuf_iterator`, 1423
  - `std::iterator`, 1425
  - `std::ostream_iterator`, 1628
  - `std::ostreambuf_iterator`, 1631
  - `std::raw_storage_iterator`, 1663
  - `std::set`, 1686
  - `std::unordered_map`, 1763
  - `std::unordered_multimap`, 1783
  - `std::unordered_multiset`, 1804
  - `std::unordered_set`, 1824
- `regex`
  - Regular Expressions, 145
- `regex.h`, 2017
- `regex_compiler.h`, 2020
- `regex_constants.h`, 2021
- `regex_cursor.h`, 2022
- `regex_error`
  - `std::regex_error`, 1664
- `regex_error.h`, 2022
- `regex_grep_matcher.h`, 2023
- `regex_iterator`
  - `std::regex_iterator`, 1665
- `regex_match`
  - Regular Expressions, 166, 168, 169
- `regex_nfa.h`, 2024
- `regex_replace`
  - Regular Expressions, 170
- `regex_search`
  - Regular Expressions, 171–173
- `regex_token_iterator`
  - `std::regex_token_iterator`, 1668, 1670
- `regex_traits`
  - `std::regex_traits`, 1672
- `register_callback`
  - `std::basic_ios`, 1116
  - `std::ios_base`, 1413
- Regular Expressions, 140
  - `__match_flag`, 146
  - `__syntax_option`, 146
  - `awk`, 175
  - `basic`, 175
  - `collate`, 175



- cregex\_token\_iterator, 145
- csub\_match, 145
- ECMAScript, 175
- egrep, 175
- error\_backref, 146
- error\_badbrace, 146
- error\_badrepeat, 147
- error\_brace, 147
- error\_brack, 147
- error\_collate, 147
- error\_complexity, 147
- error\_ctype, 147
- error\_escape, 147
- error\_paren, 147
- error\_range, 147
- error\_space, 147
- error\_stack, 147
- error\_type, 146
- extended, 175
- format\_default, 176
- format\_first\_only, 176
- format\_no\_copy, 176
- format\_sed, 176
- grep, 176
- icase, 176
- isctype, 147
- match\_any, 177
- match\_continuous, 177
- match\_default, 177
- match\_flag\_type, 145
- match\_not BOL, 177
- match\_not\_BOW, 177
- match\_not\_EOL, 177
- match\_not\_EOW, 177
- match\_not\_null, 177
- match\_prev\_avail, 177
- nosubs, 178
- operator!=, 148, 149, 151
- operator<, 151, 153, 154
- operator<<, 154
- operator<=, 154–156
- operator>, 160, 161, 163
- operator>=, 163, 164, 166
- operator==, 156, 158–160
- optimize, 178
- regex, 145
- regex\_match, 166, 168, 169
- regex\_replace, 170
- regex\_search, 171–173
- sregex\_token\_iterator, 145
- ssub\_match, 145
- swap, 174
- syntax\_option\_type, 145
- value, 174
- wcregex\_token\_iterator, 145
- wcsub\_match, 146
- wregex, 146
- wsregex\_token\_iterator, 146
- wssub\_match, 146
- rehash
  - std::unordered\_map, 1777
  - std::unordered\_multimap, 1798
  - std::unordered\_multiset, 1819
  - std::unordered\_set, 1839
- release
  - std::auto\_ptr, 1098
- remove
  - Mutating, 81
  - std::forward\_list, 1359
  - std::list, 1447
- remove\_copy
  - Mutating, 81
- remove\_copy\_if
  - Mutating, 81
- remove\_if
  - Mutating, 82
  - std::forward\_list, 1361
  - std::list, 1447
- rend
  - \_\_gnu\_cxx::\_\_versa\_string, 498
  - std::\_\_detail::\_\_Nfa, 1044
  - std::basic\_string, 1167
  - std::deque, 1322
  - std::list, 1449
  - std::map, 1483
  - std::multimap, 1558
  - std::multiset, 1576
  - std::set, 1701
  - std::vector, 1857
- replace
  - \_\_gnu\_cxx::\_\_versa\_string, 498, 500–504
  - Mutating, 82
  - std::basic\_string, 1167–1171
- replace\_copy
  - std, 404
- replace\_copy\_if
  - Mutating, 83
- replace\_if
  - Mutating, 83
- replace\_minimal\_n
  - \_\_gnu\_parallel::\_\_Settings, 672
- requested\_size
  - std::Temporary\_buffer, 1085
- reserve
  - \_\_gnu\_cxx::\_\_versa\_string, 504
  - std::\_\_detail::\_\_Nfa, 1044
  - std::basic\_string, 1173
  - std::unordered\_map, 1778



- std::unordered\_multimap, 1798
- std::unordered\_multiset, 1819
- std::unordered\_set, 1839
- std::vector, 1857
- reset
  - std::auto\_ptr, 1099
  - std::bernoulli\_distribution, 1178
  - std::binomial\_distribution, 1189
  - std::cauchy\_distribution, 1192
  - std::chi\_squared\_distribution, 1199
  - std::discrete\_distribution, 1329
  - std::exponential\_distribution, 1337
  - std::extreme\_value\_distribution, 1340
  - std::fisher\_f\_distribution, 1344
  - std::gamma\_distribution, 1370
  - std::geometric\_distribution, 1373
  - std::lognormal\_distribution, 1465
  - std::negative\_binomial\_distribution, 1582
  - std::normal\_distribution, 1586
  - std::piecewise\_constant\_distribution, 1639
  - std::piecewise\_linear\_distribution, 1644
  - std::poisson\_distribution, 1653
  - std::student\_t\_distribution, 1720
  - std::uniform\_int\_distribution, 1752
  - std::uniform\_real\_distribution, 1755
  - std::weibull\_distribution, 1865
- resize
  - \_\_gnu\_cxx::\_versa\_string, 505
  - \_\_gnu\_pbds::hash\_standard\_resize\_policy, 884
  - std::\_detail::\_Nfa, 1044, 1045
  - std::basic\_string, 1173
  - std::deque, 1322
  - std::forward\_list, 1361
  - std::list, 1449
  - std::vector, 1858
- resize\_fn\_imps.hpp, 2024, 2025
- resize\_no\_store\_hash\_fn\_imps.hpp, 2025
- resize\_policy.hpp, 2025
- resize\_store\_hash\_fn\_imps.hpp, 2025
- result\_type
  - \_\_gnu\_cxx::\_detail::\_Ffit\_finder, 440
  - \_\_gnu\_parallel::\_EqualFromLess, 624
  - \_\_gnu\_parallel::\_EqualTo, 625
  - \_\_gnu\_parallel::\_Less, 632
  - \_\_gnu\_parallel::\_Lexicographic, 634
  - \_\_gnu\_parallel::\_LexicographicReverse, 635
  - \_\_gnu\_parallel::\_Multiplies, 656
  - \_\_gnu\_parallel::\_Plus, 659
  - \_\_gnu\_parallel::\_binder1st, 589
  - \_\_gnu\_parallel::\_binder2nd, 591
  - \_\_gnu\_parallel::\_unary\_negate, 619
  - std::bernoulli\_distribution, 1177
  - std::binary\_function, 1181
  - std::binary\_negate, 1183
  - std::binder1st, 1184
  - std::binder2nd, 1186
  - std::binomial\_distribution, 1187
  - std::cauchy\_distribution, 1191
  - std::chi\_squared\_distribution, 1198
  - std::const\_mem\_fun1\_ref\_t, 1232
  - std::const\_mem\_fun1\_t, 1234
  - std::const\_mem\_fun\_ref\_t, 1235
  - std::const\_mem\_fun\_t, 1236
  - std::discard\_block\_engine, 1324
  - std::discrete\_distribution, 1328
  - std::divides, 1333
  - std::equal\_to, 1334
  - std::exponential\_distribution, 1336
  - std::extreme\_value\_distribution, 1339
  - std::fisher\_f\_distribution, 1342
  - std::gamma\_distribution, 1369
  - std::geometric\_distribution, 1372
  - std::greater, 1375
  - std::greater\_equal, 1376
  - std::hash< \_\_gnu\_cxx::throw\_value\_limit >, 1381
  - std::hash< \_\_gnu\_cxx::throw\_value\_random >, 1382
  - std::independent\_bits\_engine, 1397
  - std::less, 1427
  - std::less\_equal, 1429
  - std::linear\_congruential\_engine, 1430
  - std::logical\_and, 1461
  - std::logical\_not, 1462
  - std::logical\_or, 1463
  - std::lognormal\_distribution, 1464
  - std::mem\_fun1\_ref\_t, 1497
  - std::mem\_fun1\_t, 1498
  - std::mem\_fun\_ref\_t, 1500
  - std::mem\_fun\_t, 1501
  - std::mersenne\_twister\_engine, 1503
  - std::minus, 1511
  - std::modulus, 1512
  - std::multiplies, 1560
  - std::negate, 1579
  - std::negative\_binomial\_distribution, 1581
  - std::normal\_distribution, 1585
  - std::not\_equal\_to, 1588
  - std::owner\_less< shared\_ptr< \_Tp > >, 1634
  - std::owner\_less< weak\_ptr< \_Tp > >, 1635
  - std::piecewise\_constant\_distribution, 1638
  - std::piecewise\_linear\_distribution, 1642
  - std::plus, 1646
  - std::pointer\_to\_binary\_function, 1647
  - std::pointer\_to\_unary\_function, 1649
  - std::poisson\_distribution, 1652
  - std::random\_device, 1662
  - std::seed\_seq, 1682
  - std::shuffle\_order\_engine, 1710

- std::student\_t\_distribution, 1719
- std::unary\_function, 1749
- std::unary\_negate, 1750
- std::uniform\_int\_distribution, 1751
- std::uniform\_real\_distribution, 1754
- std::weibull\_distribution, 1864
- rethrow\_exception
  - Exceptions, 47
- rethrow\_if\_nested
  - Exceptions, 47
- return\_temporary\_buffer
  - std, 404
- reverse
  - Mutating, 83
  - std::forward\_list, 1361
  - std::list, 1449
- reverse\_copy
  - Mutating, 85
- reverse\_iteration
  - Traits, 213
- reverse\_iterator
  - std::reverse\_iterator, 1679
  - std::set, 1686
- rfind
  - \_\_gnu\_cxx::\_\_versa\_string, 505, 506
  - std::basic\_string, 1174, 1175
- right
  - std, 404
  - std::basic\_ios, 1122
  - std::ios\_base, 1417
- ropeimpl.h, 2026
- rotate
  - Mutating, 85
- rotate\_copy
  - Mutating, 86
- rotate\_fn\_imps.hpp, 2026
- SGL, 179
- safe\_base.h, 2026
- safe\_iterator.h, 2027
- safe\_local\_iterator.h, 2028
- safe\_sequence.h, 2028
- safe\_unordered\_base.h, 2029
- safe\_unordered\_container.h, 2029
- sample\_probe\_fn
  - \_\_gnu\_pbds::sample\_probe\_fn, 902
- sample\_probe\_fn.hpp, 2029
- sample\_range\_hashing
  - \_\_gnu\_pbds::sample\_range\_hashing, 903
  - \_\_gnu\_pbds::sample\_resize\_policy, 907
  - \_\_gnu\_pbds::sample\_resize\_trigger, 909
  - \_\_gnu\_pbds::sample\_size\_policy, 910
- sample\_range\_hashing.hpp, 2030
- sample\_ranged\_hash\_fn
  - \_\_gnu\_pbds::sample\_ranged\_hash\_fn, 904
- sample\_ranged\_hash\_fn.hpp, 2030
- sample\_ranged\_probe\_fn.hpp, 2030
- sample\_resize\_policy
  - \_\_gnu\_pbds::sample\_resize\_policy, 906
- sample\_resize\_policy.hpp, 2031
- sample\_resize\_trigger
  - \_\_gnu\_pbds::sample\_resize\_trigger, 908
- sample\_resize\_trigger.hpp, 2031
- sample\_size\_policy
  - \_\_gnu\_pbds::sample\_size\_policy, 910
- sample\_size\_policy.hpp, 2031
- sample\_tree\_node\_update.hpp, 2032
- sample\_trie\_access\_traits.hpp, 2032
- sample\_trie\_node\_update
  - \_\_gnu\_pbds::sample\_trie\_node\_update, 912
- sample\_trie\_node\_update.hpp, 2032
- sample\_update\_policy
  - \_\_gnu\_pbds::sample\_update\_policy, 913
- sample\_update\_policy.hpp, 2032
- scan\_is
  - std::\_\_ctype\_abstract\_base, 980
  - std::ctype, 1246
  - std::ctype< char >, 1257
  - std::ctype< wchar\_t >, 1271
  - std::ctype\_byname, 1286
  - std::ctype\_byname< char >, 1296
- scan\_not
  - std::\_\_ctype\_abstract\_base, 980
  - std::ctype, 1246
  - std::ctype< char >, 1257
  - std::ctype< wchar\_t >, 1272
  - std::ctype\_byname, 1286
  - std::ctype\_byname< char >, 1296
- scientific
  - std, 405
  - std::basic\_ios, 1122
  - std::ios\_base, 1418
- search
  - Non-Mutating, 105
- search.h, 2033
- search\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 673
- search\_n
  - Non-Mutating, 106
- second
  - \_\_gnu\_parallel::\_IteratorPair, 629
  - std::pair, 1637
  - std::sub\_match, 1724
- second\_argument\_type
  - \_\_gnu\_parallel::\_EqualFromLess, 624
  - \_\_gnu\_parallel::\_EqualTo, 625
  - \_\_gnu\_parallel::\_Less, 632
  - \_\_gnu\_parallel::\_Lexicographic, 634

- `__gnu_parallel::_LexicographicReverse`, 635
- `__gnu_parallel::_Multiplies`, 657
- `__gnu_parallel::_Plus`, 659
- `std::binary_function`, 1181
- `std::binary_negate`, 1183
- `std::const_mem_fun1_ref_t`, 1232
- `std::const_mem_fun1_t`, 1234
- `std::divides`, 1333
- `std::equal_to`, 1334
- `std::greater`, 1375
- `std::greater_equal`, 1376
- `std::less`, 1427
- `std::less_equal`, 1429
- `std::logical_and`, 1461
- `std::logical_or`, 1463
- `std::mem_fun1_ref_t`, 1497
- `std::mem_fun1_t`, 1498
- `std::minus`, 1511
- `std::modulus`, 1512
- `std::multiplies`, 1560
- `std::not_equal_to`, 1588
- `std::owner_less< shared_ptr< _Tp > >`, 1634
- `std::owner_less< weak_ptr< _Tp > >`, 1635
- `std::plus`, 1646
- `std::pointer_to_binary_function`, 1648
- `second_type`
  - `__gnu_parallel::_IteratorPair`, 629
  - `std::pair`, 1636
  - `std::sub_match`, 1723
- `seed`
  - `std::discard_block_engine`, 1326
  - `std::independent_bits_engine`, 1399
  - `std::linear_congruential_engine`, 1431
  - `std::shuffle_order_engine`, 1713
- `seed_seq`
  - `std::seed_seq`, 1682
- `seekdir`
  - `std::basic_ios`, 1108
  - `std::ios_base`, 1410
- `select_on_container_copy_construction`
  - `__gnu_cxx::_alloc_traits`, 437
  - `std::allocator_traits`, 1094
- `Sequences`, 180
- `sequential`
  - `__gnu_parallel`, 254
- `set`
  - `__gnu_parallel::_Settings`, 669
  - `std::set`, 1686, 1688, 1689
- `Set Operation`, 181
  - includes, 182, 183
  - `set_difference`, 183, 184
  - `set_intersection`, 184, 185
  - `set_symmetric_difference`, 185
  - `set_union`, 187
- `set.h`, 2033, 2034
- `set_difference`
  - Set Operation, 183, 184
- `set_difference_minimal_n`
  - `__gnu_parallel::_Settings`, 673
- `set_intersection`
  - Set Operation, 184, 185
- `set_intersection_minimal_n`
  - `__gnu_parallel::_Settings`, 673
- `set_load`
  - `__gnu_pbds::cc_hash_max_collision_check_↔`  
resize\_trigger, 709
- `set_loads`
  - `__gnu_pbds::hash_load_check_resize_trigger`, 880
- `set_num_threads`
  - `__gnu_parallel::balanced_quicksort_tag`, 676
  - `__gnu_parallel::balanced_tag`, 677
  - `__gnu_parallel::default_parallel_tag`, 679
  - `__gnu_parallel::exact_tag`, 680
  - `__gnu_parallel::multiway_mergesort_exact_tag`, 683
  - `__gnu_parallel::multiway_mergesort_sampling_tag`,  
684
  - `__gnu_parallel::multiway_mergesort_tag`, 685
  - `__gnu_parallel::omp_loop_static_tag`, 687
  - `__gnu_parallel::omp_loop_tag`, 689
  - `__gnu_parallel::parallel_tag`, 693
  - `__gnu_parallel::quicksort_tag`, 694
  - `__gnu_parallel::sampling_tag`, 695
  - `__gnu_parallel::unbalanced_tag`, 698
- `set_operations.h`, 2034
- `set_symmetric_difference`
  - Set Operation, 185
- `set_symmetric_difference_minimal_n`
  - `__gnu_parallel::_Settings`, 673
- `set_union`
  - Set Operation, 187
- `set_union_minimal_n`
  - `__gnu_parallel::_Settings`, 673
- `setf`
  - `std::basic_ios`, 1117
  - `std::ios_base`, 1413
- `setstate`
  - `std::basic_ios`, 1117
- `settings.h`, 2035
  - `_GLIBCXX_PARALLEL_CONDITION`, 2036
- `shared_ptr`
  - `std::shared_ptr`, 1704–1707
- `shared_ptr.h`, 2036
- `shared_ptr_base.h`, 2038
- `showbase`
  - `std`, 405
  - `std::basic_ios`, 1122
  - `std::ios_base`, 1418
- `showpoint`

- std, 405
- std::basic\_ios, 1122
- std::ios\_base, 1418
- showpos
  - std, 405
  - std::basic\_ios, 1123
  - std::ios\_base, 1418
- shrink\_to\_fit
  - \_\_gnu\_cxx::\_\_versa\_string, 507
  - std::\_\_detail::\_\_Nfa, 1045
  - std::basic\_string, 1175
  - std::deque, 1322
  - std::vector, 1858
- shuffle
  - Mutating, 86
- shuffle\_order\_engine
  - std::shuffle\_order\_engine, 1710, 1712
- size
  - \_\_gnu\_cxx::\_\_versa\_string, 507
  - Numeric\_arrays, 121
  - std::\_\_detail::\_\_Nfa, 1045
  - std::Temporary\_buffer, 1085
  - std::basic\_string, 1175
  - std::deque, 1322
  - std::list, 1449
  - std::map, 1483
  - std::match\_results, 1495
  - std::multimap, 1558
  - std::multiset, 1576
  - std::priority\_queue, 1657
  - std::queue, 1660
  - std::set, 1701
  - std::stack, 1718
  - std::unordered\_map, 1778
  - std::unordered\_multimap, 1800
  - std::unordered\_multiset, 1819
  - std::unordered\_set, 1839
  - std::vector, 1858
- size\_fn\_imps.hpp, 2039
- size\_type
  - \_\_gnu\_pbds::hash\_prime\_size\_policy, 881
  - \_\_gnu\_pbds::sample\_range\_hashing, 903
  - \_\_gnu\_pbds::sample\_resize\_policy, 905
  - \_\_gnu\_pbds::sample\_resize\_trigger, 908
  - \_\_gnu\_pbds::sample\_size\_policy, 910
  - \_\_gnu\_pbds::trie\_prefix\_search\_node\_update, 930
  - std::allocator\_traits, 1091
  - std::set, 1686
  - std::unordered\_map, 1763
  - std::unordered\_multimap, 1783
  - std::unordered\_multiset, 1804
  - std::unordered\_set, 1824
- skipws
  - std, 405
- std::basic\_ios, 1123
- std::ios\_base, 1418
- slice
  - Numeric\_arrays, 115
- slice\_array
  - Numeric\_arrays, 115
- slice\_array.h, 2039
- sort
  - Sorting, 207
  - std::forward\_list, 1361, 1362
  - std::list, 1450
- sort.h, 2040
- sort\_heap
  - Heap, 56, 58
- sort\_minimal\_n
  - \_\_gnu\_parallel::Settings, 673
- sort\_mwms\_oversampling
  - \_\_gnu\_parallel::Settings, 673
- sort\_qs\_num\_samples\_preset
  - \_\_gnu\_parallel::Settings, 673
- sort\_qsb\_base\_case\_maximal\_n
  - \_\_gnu\_parallel::Settings, 673
- Sorting, 189
  - inplace\_merge, 191
  - is\_sorted, 191, 193
  - is\_sorted\_until, 193
  - lexicographical\_compare, 194
  - max, 194, 196
  - max\_element, 196
  - merge, 198
  - min, 199
  - min\_element, 199, 200
  - minmax, 200
  - minmax\_element, 200, 201
  - next\_permutation, 201
  - nth\_element, 203
  - partial\_sort, 204
  - partial\_sort\_copy, 204, 206
  - prev\_permutation, 206, 207
  - sort, 207
  - stable\_sort, 208
- splay\_fn\_imps.hpp, 2040
- splay\_tree\_.hpp, 2040
- splice
  - std::list, 1450
- splice\_after
  - std::forward\_list, 1362
- split\_fn\_imps.hpp, 2041
- split\_join\_can\_throw
  - Traits, 213
- split\_join\_fn\_imps.hpp, 2041, 2042
- sregex\_token\_iterator
  - Regular Expressions, 145
- sso\_string\_base.h, 2042

- ssub\_match
  - Regular Expressions, 145
- stable\_partition
  - Mutating, 86
- stable\_sort
  - Sorting, 208
- stack
  - std::stack, 1717
- standard\_policies.hpp, 2042
- start
  - Numeric\_arrays, 121, 122
- state
  - std::fpos, 1364, 1365
- static\_pointer\_cast
  - std, 405
- std, 299
  - \_Construct, 363
  - \_Destroy, 363, 364
  - \_\_final\_insertion\_sort, 354
  - \_\_find, 354, 355
  - \_\_find\_if, 355
  - \_\_find\_if\_not, 355
  - \_\_find\_if\_not\_n, 355
  - \_\_gcd, 356
  - \_\_heap\_select, 356
  - \_\_inplace\_stable\_partition, 356
  - \_\_inplace\_stable\_sort, 356
  - \_\_insertion\_sort, 356, 357
  - \_\_introsort\_loop, 357
  - \_\_lg, 357
  - \_\_merge\_adaptive, 357
  - \_\_merge\_without\_buffer, 358
  - \_\_move\_median\_to\_first, 358
  - \_\_move\_merge, 358
  - \_\_move\_merge\_adaptive, 359
  - \_\_move\_merge\_adaptive\_backward, 359
  - \_\_partition, 359
  - \_\_reverse, 360
  - \_\_rotate, 360
  - \_\_rotate\_adaptive, 360
  - \_\_search\_n, 360, 361
  - \_\_stable\_partition\_adaptive, 361
  - \_\_umap\_traits, 353
  - \_\_ummap\_traits, 353
  - \_\_umset\_traits, 353
  - \_\_unguarded\_insertion\_sort, 361
  - \_\_unguarded\_linear\_insert, 362
  - \_\_unguarded\_partition, 362
  - \_\_unguarded\_partition\_pivot, 362
  - \_\_unique\_copy, 363
  - \_\_uset\_traits, 353
  - accumulate, 364
  - adjacent\_difference, 365
  - advance, 365
  - begin, 367
  - boolalpha, 367
  - const\_pointer\_cast, 368
  - dec, 368
  - distance, 368
  - dynamic\_pointer\_cast, 368
  - end, 368, 369
  - fixed, 369
  - get\_temporary\_buffer, 369
  - getline, 369, 371
  - hex, 373
  - inner\_product, 373
  - internal, 374
  - iota, 374
  - isalnum, 374
  - isalpha, 374
  - isctrl, 374
  - isdigit, 374
  - isgraph, 375
  - islower, 375
  - isprint, 375
  - ispunct, 375
  - isspace, 375
  - isupper, 375
  - isxdigit, 375
  - left, 375
  - noboolalpha, 375
  - noshowbase, 375
  - noshowpoint, 376
  - noshowpos, 376
  - noskipws, 376
  - nounitbuf, 376
  - nouppercase, 376
  - oct, 376
  - operator!=, 376–378
  - operator<, 382–384, 386, 387
  - operator<<, 387
  - operator<=, 388–390
  - operator>, 396, 397, 399
  - operator>>, 402, 403
  - operator>=, 399, 400, 402
  - operator+, 378, 380
  - operator==, 391–394, 396
  - partial\_sum, 403
  - replace\_copy, 404
  - return\_temporary\_buffer, 404
  - right, 404
  - scientific, 405
  - showbase, 405
  - showpoint, 405
  - showpos, 405
  - skipws, 405
  - static\_pointer\_cast, 405
  - streamoff, 353

- streampos, 353
- streamsize, 354
- swap, 405, 406
- tolower, 408
- toupper, 408
- u16streampos, 354
- u32streampos, 354
- uninitialized\_copy, 408
- uninitialized\_copy\_n, 408
- uninitialized\_fill, 409
- uninitialized\_fill\_n, 409
- unitbuf, 409
- uppercase, 409
- wstreampos, 354
- std::\_\_atomic\_base<\_ITp>, 964
- std::\_\_atomic\_base<\_PTp\*>, 966
- std::\_\_atomic\_flag\_base, 967
- std::\_\_codecvt\_abstract\_base
  - do\_out, 969
  - in, 970
  - out, 970
  - unshift, 971
- std::\_\_codecvt\_abstract\_base<\_InternT, \_ExternT, \_↵  
StateT>, 968
- std::\_\_ctype\_abstract\_base
  - char\_type, 974
  - do\_is, 974
  - do\_narrow, 974, 975
  - do\_scan\_is, 975
  - do\_scan\_not, 976
  - do\_tolower, 976
  - do\_toupper, 977
  - do\_widen, 978
  - is, 978, 979
  - narrow, 979
  - scan\_is, 980
  - scan\_not, 980
  - tolower, 981
  - toupper, 981, 982
  - widen, 982
- std::\_\_ctype\_abstract\_base<\_CharT>, 972
- std::\_\_debug, 409
- std::\_\_debug::map
  - \_M\_attach, 985
  - \_M\_attach\_single, 985
  - \_M\_const\_iterators, 987
  - \_M\_detach, 985
  - \_M\_detach\_all, 986
  - \_M\_detach\_single, 986
  - \_M\_detach\_singular, 986
  - \_M\_get\_mutex, 986
  - \_M\_invalidate\_all, 986
  - \_M\_invalidate\_if, 986
  - \_M\_iterators, 987
  - \_M\_revalidate\_singular, 986
  - \_M\_swap, 986
  - \_M\_transfer\_from\_if, 986
  - \_M\_version, 987
- std::\_\_debug::multimap
  - \_M\_attach, 990
  - \_M\_attach\_single, 990
  - \_M\_const\_iterators, 991
  - \_M\_detach, 990
  - \_M\_detach\_all, 990
  - \_M\_detach\_single, 990
  - \_M\_detach\_singular, 990
  - \_M\_get\_mutex, 990
  - \_M\_invalidate\_all, 990
  - \_M\_invalidate\_if, 990
  - \_M\_iterators, 991
  - \_M\_revalidate\_singular, 990
  - \_M\_swap, 990
  - \_M\_transfer\_from\_if, 991
  - \_M\_version, 991
- std::\_\_debug::multimap<\_Key, \_Tp, \_Compare, \_Allocator>, 983
- std::\_\_debug::multimap<\_Key, \_Tp, \_Compare, \_↵  
Allocator>, 987
- std::\_\_debug::multiset
  - \_M\_attach, 994
  - \_M\_attach\_single, 994
  - \_M\_const\_iterators, 995
  - \_M\_detach, 994
  - \_M\_detach\_all, 994
  - \_M\_detach\_single, 994
  - \_M\_detach\_singular, 994
  - \_M\_get\_mutex, 995
  - \_M\_invalidate\_all, 995
  - \_M\_invalidate\_if, 995
  - \_M\_iterators, 995
  - \_M\_revalidate\_singular, 995
  - \_M\_swap, 995
  - \_M\_transfer\_from\_if, 995
  - \_M\_version, 995
- std::\_\_debug::multiset<\_Key, \_Compare, \_Allocator>, 992
- std::\_\_debug::set
  - \_M\_attach, 998
  - \_M\_attach\_single, 998
  - \_M\_const\_iterators, 999
  - \_M\_detach, 998
  - \_M\_detach\_all, 998
  - \_M\_detach\_single, 998
  - \_M\_detach\_singular, 998
  - \_M\_get\_mutex, 999
  - \_M\_invalidate\_all, 999
  - \_M\_invalidate\_if, 999
  - \_M\_iterators, 999

[\\_M\\_revalidate\\_singular, 999](#)  
[\\_M\\_swap, 999](#)  
[\\_M\\_transfer\\_from\\_if, 999](#)  
[\\_M\\_version, 999](#)  
[std::\\_\\_debug::set<\\_Key, \\_Compare, \\_Allocator >, 996](#)  
[std::\\_\\_detail, 411](#)  
[std::\\_\\_detail::Automaton, 1000](#)  
[std::\\_\\_detail::Before\\_begin<\\_NodeAlloc >, 1000](#)  
[std::\\_\\_detail::CharMatcher<\\_InIterT, \\_TraitsT >, 1001](#)  
[std::\\_\\_detail::Compiler<\\_InIter, \\_TraitsT >, 1001](#)  
[std::\\_\\_detail::Default\\_ranged\\_hash, 1002](#)  
[std::\\_\\_detail::EndTagger<\\_FwdIterT, \\_TraitsT >, 1002](#)  
[std::\\_\\_detail::Equal\\_helper<\\_Key, \\_Value, \\_ExtractKey, \\_Equal, \\_HashCodeType, \\_\\_cache\\_hash\\_code >, 1003](#)  
[std::\\_\\_detail::Equal\\_helper<\\_Key, \\_Value, \\_ExtractKey, \\_Equal, \\_HashCodeType, false >, 1003](#)  
[std::\\_\\_detail::Equal\\_helper<\\_Key, \\_Value, \\_ExtractKey, \\_Equal, \\_HashCodeType, true >, 1003](#)  
[std::\\_\\_detail::Equality<\\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits, \\_Unique\\_keys >, 1004](#)  
[std::\\_\\_detail::Equality<\\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits, false >, 1005](#)  
[std::\\_\\_detail::Equality<\\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits, true >, 1006](#)  
[std::\\_\\_detail::Equality\\_base, 1006](#)  
[std::\\_\\_detail::Grep\\_matcher, 1007](#)  
[std::\\_\\_detail::Hash\\_code\\_base<\\_Key, \\_Value, \\_ExtractKey, \\_H1, \\_H2, \\_Default\\_ranged\\_hash, false >, 1008](#)  
[std::\\_\\_detail::Hash\\_code\\_base<\\_Key, \\_Value, \\_ExtractKey, \\_H1, \\_H2, \\_Default\\_ranged\\_hash, true >, 1009](#)  
[std::\\_\\_detail::Hash\\_code\\_base<\\_Key, \\_Value, \\_ExtractKey, \\_H1, \\_H2, \\_Hash, \\_\\_cache\\_hash\\_code >, 1007](#)  
[std::\\_\\_detail::Hash\\_code\\_base<\\_Key, \\_Value, \\_ExtractKey, \\_H1, \\_H2, \\_Hash, false >, 1010](#)  
[std::\\_\\_detail::Hash\\_node<\\_Value, \\_Cache\\_hash\\_code >, 1011](#)  
[std::\\_\\_detail::Hash\\_node<\\_Value, false >, 1012](#)  
[std::\\_\\_detail::Hash\\_node<\\_Value, true >, 1013](#)  
[std::\\_\\_detail::Hash\\_node\\_base, 1014](#)  
[std::\\_\\_detail::Hashtable\\_base<\\_Key, \\_Value, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_Traits >, 1015](#)  
[std::\\_\\_detail::Hashtable\\_ebo\\_helper<\\_Nm, \\_Tp, \\_\\_use\\_ebo >, 1016](#)  
[std::\\_\\_detail::Hashtable\\_ebo\\_helper<\\_Nm, \\_Tp, false >, 1016](#)  
[std::\\_\\_detail::Hashtable\\_ebo\\_helper<\\_Nm, \\_Tp, true >, 1017](#)  
[std::\\_\\_detail::Hashtable\\_traits<\\_Cache\\_hash\\_code, \\_\\_Constant\\_iterators, \\_Unique\\_keys >, 1017](#)  
[std::\\_\\_detail::Insert<\\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_\\_Traits, \\_\\_Constant\\_iterators, \\_Unique\\_keys >, 1018](#)  
[std::\\_\\_detail::Insert<\\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_\\_Traits, false, \\_Unique\\_keys >, 1019](#)  
[std::\\_\\_detail::Insert<\\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_\\_Traits, true, false >, 1020](#)  
[std::\\_\\_detail::Insert<\\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_\\_Traits, true, true >, 1021](#)  
[std::\\_\\_detail::Insert\\_base<\\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits >, 1023](#)  
[std::\\_\\_detail::List\\_node\\_base, 1024](#)  
[std::\\_\\_detail::Local\\_const\\_iterator<\\_Key, \\_Value, \\_ExtractKey, \\_H1, \\_H2, \\_Hash, \\_\\_constant\\_iterators, \\_\\_cache >, 1025](#)  
[std::\\_\\_detail::Local\\_iterator<\\_Key, \\_Value, \\_ExtractKey, \\_H1, \\_H2, \\_Hash, \\_\\_constant\\_iterators, \\_\\_cache >, 1026](#)  
[std::\\_\\_detail::Local\\_iterator\\_base<\\_Key, \\_Value, \\_ExtractKey, \\_H1, \\_H2, \\_Hash, \\_\\_cache\\_hash\\_code >, 1027](#)  
[std::\\_\\_detail::Local\\_iterator\\_base<\\_Key, \\_Value, \\_ExtractKey, \\_H1, \\_H2, \\_Hash, false >, 1027](#)  
[std::\\_\\_detail::Local\\_iterator\\_base<\\_Key, \\_Value, \\_ExtractKey, \\_H1, \\_H2, \\_Hash, true >, 1029](#)  
[std::\\_\\_detail::Map\\_base<\\_Key, \\_Pair, \\_Alloc, \\_Select1st, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits, false >, 1030](#)  
[std::\\_\\_detail::Map\\_base<\\_Key, \\_Pair, \\_Alloc, \\_Select1st, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits, true >, 1031](#)  
[std::\\_\\_detail::Map\\_base<\\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits, \\_Unique\\_keys >, 1030](#)  
[std::\\_\\_detail::Mod\\_range\\_hashing, 1031](#)  
[std::\\_\\_detail::Nfa, 1032](#)  
[\\_M\\_allocate\\_and\\_copy, 1035](#)  
[\\_M\\_range\\_check, 1035](#)  
[assign, 1035, 1036](#)  
[at, 1036](#)  
[back, 1037](#)  
[begin, 1037](#)  
[capacity, 1037](#)  
[cbegin, 1037](#)  
[cend, 1037](#)  
[clear, 1038](#)  
[crbegin, 1038](#)



- crend, [1038](#)
- data, [1038](#)
- emplace, [1038](#)
- empty, [1039](#)
- end, [1039](#)
- erase, [1039](#)
- front, [1040](#)
- insert, [1040](#), [1042](#)
- max\_size, [1042](#)
- operator[], [1042](#), [1043](#)
- pop\_back, [1043](#)
- push\_back, [1043](#)
- rbegin, [1043](#), [1044](#)
- rend, [1044](#)
- reserve, [1044](#)
- resize, [1044](#), [1045](#)
- shrink\_to\_fit, [1045](#)
- size, [1045](#)
- swap, [1045](#)
- std::\_\_detail::\_\_Node\_const\_iterator< \_\_Value, \_\_constant\_iterators, \_\_cache >, [1046](#)
- std::\_\_detail::\_\_Node\_iterator< \_\_Value, \_\_constant\_iterators, \_\_cache >, [1047](#)
- std::\_\_detail::\_\_Node\_iterator\_base< \_\_Value, \_\_Cache, \_\_hash\_code >, [1048](#)
- std::\_\_detail::\_\_PatternCursor, [1049](#)
- std::\_\_detail::\_\_Prime\_rehash\_policy, [1049](#)
- std::\_\_detail::\_\_RangeMatcher< \_\_InIterT, \_\_TraitsT >, [1050](#)
- std::\_\_detail::\_\_Rehash\_base< \_\_Key, \_\_Value, \_\_Alloc, \_\_ExtractKey, \_\_Equal, \_\_H1, \_\_H2, \_\_Hash, \_\_Prime\_rehash\_policy, \_\_Traits >, [1051](#)
- std::\_\_detail::\_\_Rehash\_base< \_\_Key, \_\_Value, \_\_Alloc, \_\_ExtractKey, \_\_Equal, \_\_H1, \_\_H2, \_\_Hash, \_\_RehashPolicy, \_\_Traits >, [1051](#)
- std::\_\_detail::\_\_Results, [1052](#)
- std::\_\_detail::\_\_Scanner< \_\_InputIterator >, [1053](#)
- std::\_\_detail::\_\_Scanner\_base, [1054](#)
- std::\_\_detail::\_\_SpecializedCursor< \_\_FwdIterT >, [1055](#)
- std::\_\_detail::\_\_SpecializedResults< \_\_FwdIterT, \_\_Alloc >, [1056](#)
- std::\_\_detail::\_\_StartTagger< \_\_FwdIterT, \_\_TraitsT >, [1056](#)
- std::\_\_detail::\_\_State, [1057](#)
- std::\_\_detail::\_\_StateSeq, [1058](#)
- std::\_\_exception\_ptr::exception\_ptr, [1058](#)
- std::\_\_has\_iterator\_category\_helper< \_\_Tp >, [1059](#)
- std::\_\_parallel, [413](#)
- std::\_\_parallel::\_\_CRandNumber< \_\_MustBeInt >, [1059](#)
- std::\_\_profile, [427](#)
- std::\_\_profile::map< \_\_Key, \_\_Tp, \_\_Compare, \_\_Allocator >, [1059](#)
- std::\_\_profile::multimap< \_\_Key, \_\_Tp, \_\_Compare, \_\_Allocator >, [1061](#)
- std::\_\_profile::multiset< \_\_Key, \_\_Compare, \_\_Allocator >, [1063](#)
- std::\_\_profile::set< \_\_Key, \_\_Compare, \_\_Allocator >, [1065](#)
- std::\_\_Deque\_base
  - \_M\_initialize\_map, [1068](#)
- std::\_\_Deque\_base< \_\_Tp, \_\_Alloc >, [1066](#)
- std::\_\_Deque\_iterator
  - \_M\_set\_node, [1070](#)
- std::\_\_Deque\_iterator< \_\_Tp, \_\_Ref, \_\_Ptr >, [1069](#)
- std::\_\_Fwd\_list\_base< \_\_Tp, \_\_Alloc >, [1070](#)
- std::\_\_Fwd\_list\_const\_iterator< \_\_Tp >, [1072](#)
- std::\_\_Fwd\_list\_iterator< \_\_Tp >, [1072](#)
- std::\_\_Fwd\_list\_node< \_\_Tp >, [1073](#)
- std::\_\_Fwd\_list\_node\_base, [1074](#)
- std::\_\_Hashtable< \_\_Key, \_\_Value, \_\_Alloc, \_\_ExtractKey, \_\_Equal, \_\_H1, \_\_H2, \_\_Hash, \_\_RehashPolicy, \_\_Traits >, [1075](#)
- std::\_\_List\_base< \_\_Tp, \_\_Alloc >, [1080](#)
- std::\_\_List\_const\_iterator< \_\_Tp >, [1081](#)
- std::\_\_List\_iterator< \_\_Tp >, [1082](#)
- std::\_\_List\_node
  - \_M\_data, [1083](#)
- std::\_\_List\_node< \_\_Tp >, [1083](#)
- std::\_\_Temporary\_buffer
  - \_Temporary\_buffer, [1084](#)
  - begin, [1085](#)
  - end, [1085](#)
  - requested\_size, [1085](#)
  - size, [1085](#)
- std::\_\_Temporary\_buffer< \_\_ForwardIterator, \_\_Tp >, [1084](#)
- std::\_\_Vector\_base< \_\_Tp, \_\_Alloc >, [1086](#)
- std::allocator< \_\_Tp >, [1087](#)
- std::allocator< void >, [1088](#)
- std::allocator\_arg\_t, [1088](#)
- std::allocator\_traits
  - allocate, [1091](#)
  - allocator\_type, [1090](#)
  - const\_pointer, [1090](#)
  - const\_void\_pointer, [1090](#)
  - construct, [1092](#)
  - deallocate, [1092](#)
  - destroy, [1092](#)
  - difference\_type, [1090](#)
  - max\_size, [1092](#)
  - pointer, [1090](#)
  - propagate\_on\_container\_copy\_assignment, [1090](#)
  - propagate\_on\_container\_move\_assignment, [1090](#)
  - propagate\_on\_container\_swap, [1091](#)
  - select\_on\_container\_copy\_construction, [1094](#)
  - size\_type, [1091](#)
  - value\_type, [1091](#)
  - void\_pointer, [1091](#)
- std::allocator\_traits< \_\_Alloc >, [1089](#)
- std::atomic\_flag, [1094](#)
- std::auto\_ptr
  - ~auto\_ptr, [1097](#)



- auto\_ptr, 1096, 1097
- element\_type, 1096
- get, 1097
- operator\*, 1098
- operator->, 1098
- operator=, 1098
- release, 1098
- reset, 1099
- std::auto\_ptr<\_Tp>, 1095
- std::auto\_ptr\_ref<\_Tp1>, 1099
- std::back\_insert\_iterator
  - back\_insert\_iterator, 1101
  - container\_type, 1101
  - difference\_type, 1101
  - iterator\_category, 1101
  - operator\*, 1101
  - operator++, 1102
  - operator=, 1102
  - pointer, 1101
  - reference, 1101
  - value\_type, 1101
- std::back\_insert\_iterator<\_Container>, 1100
- std::bad\_weak\_ptr, 1102
- std::basic\_ios
  - \_M\_getloc, 1109
  - \_\_ctype\_type, 1106
  - \_\_num\_get\_type, 1106
  - \_\_num\_put\_type, 1106
  - ~basic\_ios, 1109
  - adjustfield, 1120
  - app, 1120
  - ate, 1120
  - bad, 1110
  - badbit, 1120
  - basefield, 1120
  - basic\_ios, 1109
  - beg, 1120
  - binary, 1120
  - boolalpha, 1120
  - char\_type, 1106
  - clear, 1110
  - copyfmt, 1110
  - cur, 1120
  - dec, 1121
  - end, 1121
  - eof, 1110
  - eofbit, 1121
  - event, 1109
  - event\_callback, 1106
  - exceptions, 1111
  - fail, 1111
  - failbit, 1121
  - fill, 1112
  - fixed, 1121
  - flags, 1112
  - floatfield, 1121
  - fmtflags, 1107
  - getloc, 1113
  - good, 1113
  - goodbit, 1121
  - hex, 1121
  - imbue, 1113
  - in, 1121
  - init, 1113
  - int\_type, 1107
  - internal, 1122
  - iostate, 1107
  - isword, 1114
  - left, 1122
  - narrow, 1114
  - oct, 1122
  - off\_type, 1108
  - openmode, 1108
  - operator void \*, 1114
  - operator!, 1115
  - out, 1122
  - pos\_type, 1108
  - precision, 1115
  - pword, 1115
  - rdbuf, 1115, 1116
  - rdstate, 1116
  - register\_callback, 1116
  - right, 1122
  - scientific, 1122
  - seekdir, 1108
  - setf, 1117
  - setstate, 1117
  - showbase, 1122
  - showpoint, 1122
  - showpos, 1123
  - skipws, 1123
  - sync\_with\_stdio, 1117
  - tie, 1118
  - traits\_type, 1109
  - trunc, 1123
  - unitbuf, 1123
  - unsetf, 1118
  - uppercase, 1123
  - widen, 1118
  - width, 1119
  - xalloc, 1119
- std::basic\_ios<\_CharT, \_Traits>, 1103
- std::basic\_regex
  - ~basic\_regex, 1127
  - assign, 1127–1129
  - basic\_regex, 1125, 1126
  - flags, 1129
  - getloc, 1129

- imbue, 1129
- mark\_count, 1130
- operator=, 1130
- swap, 1130
- std::basic\_regex< \_Ch\_type, \_Rx\_traits >, 1123
- std::basic\_string
  - ~basic\_string, 1138
  - append, 1138–1140
  - assign, 1140, 1142, 1143
  - at, 1143, 1144
  - back, 1144
  - basic\_string, 1136–1138
  - begin, 1144
  - c\_str, 1144
  - capacity, 1145
  - cbegin, 1145
  - cend, 1145
  - clear, 1145
  - compare, 1145–1147
  - copy, 1147
  - crbegin, 1149
  - crend, 1149
  - data, 1149
  - empty, 1149
  - end, 1149
  - erase, 1150
  - find, 1151, 1152
  - find\_first\_not\_of, 1152, 1153
  - find\_first\_of, 1153, 1154
  - find\_last\_not\_of, 1155, 1156
  - find\_last\_of, 1156, 1157
  - front, 1157
  - get\_allocator, 1158
  - insert, 1158, 1160–1162
  - length, 1162
  - max\_size, 1162
  - npos, 1176
  - operator+=, 1162, 1164
  - operator=, 1164, 1165
  - operator[], 1165, 1166
  - pop\_back, 1166
  - push\_back, 1166
  - rbegin, 1166
  - rend, 1167
  - replace, 1167–1171
  - reserve, 1173
  - resize, 1173
  - rfind, 1174, 1175
  - shrink\_to\_fit, 1175
  - size, 1175
  - substr, 1175
  - swap, 1176
- std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1132
- std::bernoulli\_distribution, 1176
  - bernoulli\_distribution, 1177
  - max, 1178
  - min, 1178
  - operator(), 1178
  - operator==, 1178
  - p, 1178
  - param, 1178
  - reset, 1178
  - result\_type, 1177
- std::bernoulli\_distribution::param\_type, 1179
- std::bidirectional\_iterator\_tag, 1180
- std::binary\_function
  - first\_argument\_type, 1181
  - result\_type, 1181
  - second\_argument\_type, 1181
- std::binary\_function< \_Arg1, \_Arg2, \_Result >, 1181
- std::binary\_negate
  - first\_argument\_type, 1183
  - result\_type, 1183
  - second\_argument\_type, 1183
- std::binary\_negate< \_Predicate >, 1182
- std::binder1st
  - argument\_type, 1184
  - result\_type, 1184
- std::binder1st< \_Operation >, 1183
- std::binder2nd
  - argument\_type, 1185
  - result\_type, 1186
- std::binder2nd< \_Operation >, 1185
- std::binomial\_distribution
  - max, 1187
  - min, 1187
  - operator<<, 1189
  - operator>>, 1189
  - operator(), 1187
  - operator==, 1189
  - p, 1187
  - param, 1187
  - reset, 1189
  - result\_type, 1187
  - t, 1189
- std::binomial\_distribution< \_IntType >, 1186
- std::binomial\_distribution< \_IntType >::param\_type, 1190
- std::cauchy\_distribution
  - max, 1191
  - min, 1191
  - operator(), 1192
  - operator==, 1192
  - param, 1192
  - reset, 1192
  - result\_type, 1191
- std::cauchy\_distribution< \_RealType >, 1190
- std::cauchy\_distribution< \_RealType >::param\_type, 1192

std::char\_traits< \_\_gnu\_cxx::character< V, I, S > >, 1194  
 std::char\_traits< \_CharT >, 1193  
 std::char\_traits< char >, 1195  
 std::char\_traits< wchar\_t >, 1196  
 std::chi\_squared\_distribution  
     max, 1198  
     min, 1198  
     operator<<, 1199  
     operator>>, 1199  
     operator(), 1198  
     operator==, 1199  
     param, 1198  
     reset, 1199  
     result\_type, 1198  
 std::chi\_squared\_distribution< \_RealType >, 1197  
 std::chi\_squared\_distribution< \_RealType >::param\_↵  
     type, 1200  
 std::codecvt  
     do\_out, 1202  
     in, 1203  
     out, 1203  
     unshift, 1204  
 std::codecvt< \_InternT, \_ExternT, \_StateT >, 1201  
 std::codecvt< \_InternT, \_ExternT, encoding\_state >, 1205  
     do\_out, 1206  
     in, 1206  
     out, 1207  
     unshift, 1208  
 std::codecvt< char, char, mbstate\_t >, 1209  
     do\_out, 1210  
     in, 1210  
     out, 1211  
     unshift, 1212  
 std::codecvt< wchar\_t, char, mbstate\_t >, 1213  
     do\_out, 1214  
     in, 1215  
     out, 1215  
     unshift, 1216  
 std::codecvt\_base, 1217  
 std::codecvt\_byname  
     do\_out, 1219  
     in, 1220  
     out, 1220  
     unshift, 1221  
 std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >,  
     1218  
 std::collate  
     ~collate, 1225  
     char\_type, 1223  
     collate, 1223, 1225  
     compare, 1225  
     do\_compare, 1225  
     do\_hash, 1226  
     do\_transform, 1226  
     hash, 1226  
     id, 1227  
     string\_type, 1223  
     transform, 1227  
 std::collate< \_CharT >, 1222  
 std::collate\_byname  
     char\_type, 1229  
     compare, 1229  
     do\_compare, 1230  
     do\_hash, 1230  
     do\_transform, 1230  
     hash, 1230  
     id, 1231  
     string\_type, 1229  
     transform, 1231  
 std::collate\_byname< \_CharT >, 1228  
 std::const\_mem\_fun1\_ref\_t  
     first\_argument\_type, 1232  
     result\_type, 1232  
     second\_argument\_type, 1232  
 std::const\_mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >, 1232  
 std::const\_mem\_fun1\_t  
     first\_argument\_type, 1234  
     result\_type, 1234  
     second\_argument\_type, 1234  
 std::const\_mem\_fun1\_t< \_Ret, \_Tp, \_Arg >, 1233  
 std::const\_mem\_fun\_ref\_t  
     argument\_type, 1235  
     result\_type, 1235  
 std::const\_mem\_fun\_ref\_t< \_Ret, \_Tp >, 1234  
 std::const\_mem\_fun\_t  
     argument\_type, 1236  
     result\_type, 1236  
 std::const\_mem\_fun\_t< \_Ret, \_Tp >, 1235  
 std::ctype  
     do\_is, 1239  
     do\_narrow, 1239, 1240  
     do\_scan\_is, 1240  
     do\_scan\_not, 1241  
     do\_tolower, 1241  
     do\_toupper, 1242  
     do\_widen, 1242, 1243  
     id, 1248  
     is, 1243, 1244  
     narrow, 1244  
     scan\_is, 1246  
     scan\_not, 1246  
     tolower, 1246, 1247  
     toupper, 1247  
     widen, 1248  
 std::ctype< \_CharT >, 1237  
 std::ctype< char >, 1249  
     ~ctype, 1251  
     char\_type, 1251

- classic\_table, 1252
- ctype, 1251
- do\_narrow, 1252
- do\_tolower, 1252, 1253
- do\_toupper, 1253
- do\_widen, 1254
- id, 1260
- is, 1254, 1256
- narrow, 1256
- scan\_is, 1257
- scan\_not, 1257
- table, 1257
- table\_size, 1260
- tolower, 1258
- toupper, 1258, 1259
- widen, 1259
- std::ctype< wchar\_t >, 1261
  - ~ctype, 1263
  - char\_type, 1263
  - ctype, 1263
  - do\_is, 1263, 1265
  - do\_narrow, 1265
  - do\_scan\_is, 1266
  - do\_scan\_not, 1266
  - do\_tolower, 1267
  - do\_toupper, 1267, 1269
  - do\_widen, 1269
  - id, 1275
  - is, 1270
  - narrow, 1270, 1271
  - scan\_is, 1271
  - scan\_not, 1272
  - tolower, 1272
  - toupper, 1274
  - widen, 1274, 1275
- std::ctype\_base, 1276
- std::ctype\_byname
  - do\_is, 1279
  - do\_narrow, 1279, 1280
  - do\_scan\_is, 1280
  - do\_scan\_not, 1281
  - do\_tolower, 1281
  - do\_toupper, 1283
  - do\_widen, 1283, 1284
  - id, 1288
  - is, 1284
  - narrow, 1285
  - scan\_is, 1286
  - scan\_not, 1286
  - tolower, 1286, 1287
  - toupper, 1287
  - widen, 1288
- std::ctype\_byname< \_CharT >, 1277
- std::ctype\_byname< char >, 1289
- char\_type, 1291
- classic\_table, 1291
- do\_narrow, 1291
- do\_tolower, 1292
- do\_toupper, 1292, 1294
- do\_widen, 1294
- id, 1299
- is, 1295
- narrow, 1295, 1296
- scan\_is, 1296
- scan\_not, 1296
- table, 1297
- table\_size, 1299
- tolower, 1297
- toupper, 1298
- widen, 1298, 1299
- std::default\_delete< \_Tp >, 1300
- std::default\_delete< \_Tp[] >, 1300
- std::deque
  - \_M\_fill\_initialize, 1309
  - \_M\_initialize\_map, 1310
  - \_M\_new\_elements\_at\_back, 1310
  - \_M\_new\_elements\_at\_front, 1310
  - \_M\_pop\_back\_aux, 1310
  - \_M\_pop\_front\_aux, 1310
  - \_M\_push\_back\_aux, 1310
  - \_M\_push\_front\_aux, 1310
  - \_M\_range\_check, 1311
  - \_M\_range\_initialize, 1311
  - \_M\_reallocate\_map, 1311
  - \_M\_reserve\_elements\_at\_back, 1312
  - \_M\_reserve\_elements\_at\_front, 1312
  - \_M\_reserve\_map\_at\_back, 1312
  - \_M\_reserve\_map\_at\_front, 1312
  - ~deque, 1309
- assign, 1312, 1313
- at, 1313
- back, 1314
- begin, 1314
- cbegin, 1314
- cend, 1314
- clear, 1315
- crbegin, 1315
- crend, 1315
- deque, 1305, 1307, 1309
- emplace, 1315
- empty, 1315
- end, 1315, 1316
- erase, 1316
- front, 1316, 1317
- get\_allocator, 1317
- insert, 1317, 1318
- max\_size, 1318
- operator=, 1318, 1320

- operator[], 1320
- pop\_back, 1320
- pop\_front, 1321
- push\_back, 1321
- push\_front, 1321
- rbegin, 1321
- rend, 1322
- resize, 1322
- shrink\_to\_fit, 1322
- size, 1322
- swap, 1323
- std::deque<\_Tp, \_Alloc >, 1301
- std::discard\_block\_engine
  - base, 1325
  - discard, 1325
  - discard\_block\_engine, 1324, 1325
  - max, 1325
  - min, 1326
  - operator<<, 1326
  - operator>>, 1327
  - operator(), 1326
  - operator==, 1327
  - result\_type, 1324
  - seed, 1326
- std::discard\_block\_engine<\_RandomNumberEngine, \_↵  
\_p, \_r >, 1323
- std::discrete\_distribution
  - max, 1329
  - min, 1329
  - operator<<, 1329
  - operator>>, 1331
  - operator(), 1329
  - operator==, 1331
  - param, 1329
  - probabilities, 1329
  - reset, 1329
  - result\_type, 1328
- std::discrete\_distribution<\_IntType >, 1327
- std::discrete\_distribution<\_IntType >::param\_type, 1331
- std::divides
  - first\_argument\_type, 1333
  - result\_type, 1333
  - second\_argument\_type, 1333
- std::divides<\_Tp >, 1332
- std::enable\_shared\_from\_this<\_Tp >, 1333
- std::equal\_to
  - first\_argument\_type, 1334
  - result\_type, 1334
  - second\_argument\_type, 1334
- std::equal\_to<\_Tp >, 1334
- std::exponential\_distribution
  - exponential\_distribution, 1336
  - lambda, 1336
  - max, 1336
  - min, 1336
  - operator(), 1336
  - operator==, 1337
  - param, 1337
  - reset, 1337
  - result\_type, 1336
- std::exponential\_distribution<\_RealType >, 1335
- std::exponential\_distribution<\_RealType >::param\_type, 1337
- std::extreme\_value\_distribution
  - a, 1339
  - b, 1339
  - max, 1339
  - min, 1339
  - operator(), 1339
  - operator==, 1340
  - param, 1340
  - reset, 1340
  - result\_type, 1339
- std::extreme\_value\_distribution<\_RealType >, 1338
- std::extreme\_value\_distribution<\_RealType >::param\_↵  
type, 1340
- std::fisher\_f\_distribution
  - max, 1342
  - min, 1342
  - operator<<, 1344
  - operator>>, 1344
  - operator(), 1342
  - operator==, 1344
  - param, 1342
  - reset, 1344
  - result\_type, 1342
- std::fisher\_f\_distribution<\_RealType >, 1341
- std::fisher\_f\_distribution<\_RealType >::param\_type, 1345
- std::forward\_iterator\_tag, 1346
- std::forward\_list
  - ~forward\_list, 1351
  - assign, 1351, 1352
  - before\_begin, 1352
  - begin, 1352, 1353
  - cbefore\_begin, 1353
  - cbegin, 1353
  - cend, 1353
  - clear, 1353
  - emplace\_after, 1353
  - emplace\_front, 1354
  - empty, 1354
  - end, 1354
  - erase\_after, 1354, 1355
  - forward\_list, 1349–1351
  - front, 1355
  - get\_allocator, 1355
  - insert\_after, 1355, 1357

- max\_size, 1358
- merge, 1358
- operator=, 1358, 1359
- pop\_front, 1359
- push\_front, 1359
- remove, 1359
- remove\_if, 1361
- resize, 1361
- reverse, 1361
- sort, 1361, 1362
- splice\_after, 1362
- swap, 1362
- unique, 1363
- std::forward\_list< \_Tp, \_Alloc >, 1347
- std::fpos
  - fpos, 1364
  - operator streamoff, 1364
  - operator+, 1364
  - operator+=, 1364
  - operator-, 1364
  - operator-=, 1364
  - state, 1364, 1365
- std::fpos< \_StateT >, 1363
- std::front\_insert\_iterator
  - container\_type, 1366
  - difference\_type, 1366
  - front\_insert\_iterator, 1366
  - iterator\_category, 1366
  - operator\*, 1367
  - operator++, 1367
  - operator=, 1367
  - pointer, 1366
  - reference, 1366
  - value\_type, 1366
- std::front\_insert\_iterator< \_Container >, 1365
- std::gamma\_distribution
  - alpha, 1369
  - beta, 1369
  - gamma\_distribution, 1369
  - max, 1369
  - min, 1369
  - operator<<, 1370
  - operator>>, 1370
  - operator(), 1369
  - operator==, 1370
  - param, 1369, 1370
  - reset, 1370
  - result\_type, 1369
- std::gamma\_distribution< \_RealType >, 1367
- std::gamma\_distribution< \_RealType >::param\_type, 1371
- std::geometric\_distribution
  - max, 1372
  - min, 1372
  - operator(), 1373
  - operator==, 1373
  - p, 1373
  - param, 1373
  - reset, 1373
  - result\_type, 1372
- std::geometric\_distribution< \_IntType >, 1371
- std::geometric\_distribution< \_IntType >::param\_type, 1374
- std::greater
  - first\_argument\_type, 1375
  - result\_type, 1375
  - second\_argument\_type, 1375
- std::greater< \_Tp >, 1374
- std::greater\_equal
  - first\_argument\_type, 1376
  - result\_type, 1376
  - second\_argument\_type, 1376
- std::greater\_equal< \_Tp >, 1375
- std::gslice, 1376
- std::gslice\_array< \_Tp >, 1377
- std::hash< \_\_gnu\_cxx::\_\_u16vstring >, 1378
- std::hash< \_\_gnu\_cxx::\_\_u32vstring >, 1379
- std::hash< \_\_gnu\_cxx::\_\_vstring >, 1379
- std::hash< \_\_gnu\_cxx::\_\_wvstring >, 1380
- std::hash< \_\_gnu\_cxx::throw\_value\_limit >, 1381
  - argument\_type, 1381
  - result\_type, 1381
- std::hash< \_\_gnu\_cxx::throw\_value\_random >, 1382
  - argument\_type, 1382
  - result\_type, 1382
- std::hash< \_\_shared\_ptr< \_Tp, \_Lp > >, 1383
- std::hash< \_Tp >, 1378
- std::hash< \_Tp \* >, 1383
- std::hash< bool >, 1384
- std::hash< char >, 1384
- std::hash< char16\_t >, 1385
- std::hash< char32\_t >, 1385
- std::hash< double >, 1386
- std::hash< float >, 1386
- std::hash< int >, 1387
- std::hash< long >, 1387
- std::hash< long double >, 1388
- std::hash< long long >, 1388
- std::hash< shared\_ptr< \_Tp > >, 1389
- std::hash< short >, 1389
- std::hash< signed char >, 1390
- std::hash< string >, 1390
- std::hash< u16string >, 1391
- std::hash< u32string >, 1391
- std::hash< unique\_ptr< \_Tp, \_Dp > >, 1392
- std::hash< unsigned char >, 1392
- std::hash< unsigned int >, 1393
- std::hash< unsigned long >, 1393

std::hash< unsigned long long >, 1394  
 std::hash< unsigned short >, 1394  
 std::hash< wchar\_t >, 1395  
 std::hash< wstring >, 1395  
 std::hash<::vector< bool, \_Alloc > >, 1396  
 std::independent\_bits\_engine  
     base, 1398  
     discard, 1399  
     independent\_bits\_engine, 1397, 1398  
     max, 1399  
     min, 1399  
     operator>>, 1400  
     operator(), 1399  
     operator==, 1400  
     result\_type, 1397  
     seed, 1399  
 std::independent\_bits\_engine< \_RandomNumberEngine,  
     \_\_w, \_UIntType >, 1396  
 std::indirect\_array< \_Tp >, 1400  
 std::input\_iterator\_tag, 1402  
 std::insert\_iterator  
     container\_type, 1404  
     difference\_type, 1404  
     insert\_iterator, 1404  
     iterator\_category, 1404  
     operator\*, 1404  
     operator++, 1405  
     operator=, 1405  
     pointer, 1404  
     reference, 1404  
     value\_type, 1404  
 std::insert\_iterator< \_Container >, 1403  
 std::ios\_base, 1406  
     \_M\_getloc, 1411  
     ~ios\_base, 1410  
     adjustfield, 1415  
     app, 1415  
     ate, 1415  
     badbit, 1415  
     basefield, 1415  
     beg, 1416  
     binary, 1416  
     boolalpha, 1416  
     cur, 1416  
     dec, 1416  
     end, 1416  
     eofbit, 1416  
     event, 1410  
     event\_callback, 1408  
     failbit, 1416  
     fixed, 1416  
     flags, 1411  
     floatfield, 1417  
     fmtflags, 1408  
     getloc, 1411  
     goodbit, 1417  
     hex, 1417  
     imbue, 1411  
     in, 1417  
     internal, 1417  
     iostate, 1409  
     iword, 1412  
     left, 1417  
     oct, 1417  
     openmode, 1409  
     out, 1417  
     precision, 1412  
     pword, 1412  
     register\_callback, 1413  
     right, 1417  
     scientific, 1418  
     seekdir, 1410  
     setf, 1413  
     showbase, 1418  
     showpoint, 1418  
     showpos, 1418  
     skipws, 1418  
     sync\_with\_stdio, 1414  
     trunc, 1418  
     unitbuf, 1418  
     unsetf, 1414  
     uppercase, 1418  
     width, 1414  
     xalloc, 1415  
 std::ios\_base::failure, 1419  
 std::istream\_iterator  
     difference\_type, 1420  
     istream\_iterator, 1421  
     iterator\_category, 1420  
     pointer, 1420  
     reference, 1420  
     value\_type, 1421  
 std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist >, 1419  
 std::istreambuf\_iterator  
     char\_type, 1422  
     difference\_type, 1422  
     equal, 1424  
     int\_type, 1423  
     istream\_type, 1423  
     istreambuf\_iterator, 1424  
     iterator\_category, 1423  
     operator\*, 1424  
     operator++, 1424  
     pointer, 1423  
     reference, 1423  
     streambuf\_type, 1423  
     traits\_type, 1423  
     value\_type, 1423

- std::istreambuf\_iterator< \_CharT, \_Traits >, 1421
- std::iterator
  - difference\_type, 1425
  - iterator\_category, 1425
  - pointer, 1425
  - reference, 1425
  - value\_type, 1426
- std::iterator< \_Category, \_Tp, \_Distance, \_Pointer, \_↔  
Reference >, 1425
- std::iterator\_traits< \_Tp \* >, 1426
- std::iterator\_traits< const \_Tp \* >, 1426
- std::less
  - first\_argument\_type, 1427
  - result\_type, 1427
  - second\_argument\_type, 1427
- std::less< \_Tp >, 1427
- std::less\_equal
  - first\_argument\_type, 1428
  - result\_type, 1429
  - second\_argument\_type, 1429
- std::less\_equal< \_Tp >, 1428
- std::linear\_congruential\_engine
  - discard, 1431
  - increment, 1433
  - linear\_congruential\_engine, 1430
  - max, 1431
  - min, 1431
  - modulus, 1433
  - multiplier, 1433
  - operator<<, 1432
  - operator>>, 1432
  - operator(), 1431
  - operator==, 1432
  - result\_type, 1430
  - seed, 1431
- std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, ↵  
\_\_m >, 1429
- std::list
  - \_M\_create\_node, 1438
  - assign, 1439
  - back, 1439
  - begin, 1440
  - cbegin, 1440
  - cend, 1440
  - clear, 1440
  - crbegin, 1440
  - crend, 1440
  - emplace, 1441
  - empty, 1441
  - end, 1441
  - erase, 1441, 1442
  - front, 1442
  - get\_allocator, 1442
  - insert, 1442, 1444, 1445
  - list, 1437, 1438
  - max\_size, 1445
  - merge, 1445
  - operator=, 1446
  - pop\_back, 1446
  - pop\_front, 1446
  - push\_back, 1447
  - push\_front, 1447
  - rbegin, 1447
  - remove, 1447
  - remove\_if, 1447
  - rend, 1449
  - resize, 1449
  - reverse, 1449
  - size, 1449
  - sort, 1450
  - splice, 1450
  - swap, 1451
  - unique, 1451
- std::list< \_Tp, \_Alloc >, 1433
- std::locale, 1451
  - ~locale, 1454
  - all, 1456
  - category, 1453
  - classic, 1454
  - collate, 1456
  - combine, 1454
  - ctype, 1456
  - global, 1455
  - locale, 1453, 1454
  - messages, 1457
  - monetary, 1457
  - name, 1455
  - none, 1457
  - numeric, 1457
  - operator!=, 1455
  - operator(), 1455
  - operator=, 1456
  - operator==, 1456
  - time, 1457
- std::locale::facet, 1458
  - ~facet, 1459
  - facet, 1459
- std::locale::id, 1459
  - id, 1460
- std::logical\_and
  - first\_argument\_type, 1461
  - result\_type, 1461
  - second\_argument\_type, 1461
- std::logical\_and< \_Tp >, 1460
- std::logical\_not
  - argument\_type, 1462
  - result\_type, 1462
- std::logical\_not< \_Tp >, 1461



std::logical\_or  
     first\_argument\_type, 1463  
     result\_type, 1463  
     second\_argument\_type, 1463  
 std::logical\_or< \_Tp >, 1462  
 std::lognormal\_distribution  
     max, 1465  
     min, 1465  
     operator<<, 1465  
     operator>>, 1466  
     operator(), 1465  
     operator==, 1466  
     param, 1465  
     reset, 1465  
     result\_type, 1464  
 std::lognormal\_distribution< \_RealType >, 1463  
 std::lognormal\_distribution< \_RealType >::param\_type, 1466  
 std::map  
     at, 1471  
     begin, 1473  
     cbegin, 1473  
     cend, 1473  
     clear, 1473  
     count, 1474  
     crbegin, 1474  
     crend, 1474  
     emplace, 1474  
     emplace\_hint, 1475  
     empty, 1475  
     end, 1475  
     equal\_range, 1475, 1477  
     erase, 1477, 1478  
     find, 1478, 1479  
     get\_allocator, 1479  
     insert, 1479, 1480  
     key\_comp, 1480  
     lower\_bound, 1481  
     map, 1470, 1471  
     max\_size, 1481  
     operator=, 1481, 1482  
     operator[], 1482  
     rbegin, 1483  
     rend, 1483  
     size, 1483  
     swap, 1483  
     upper\_bound, 1485  
     value\_comp, 1485  
 std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 1467  
 std::mask\_array< \_Tp >, 1485  
 std::match\_results  
     ~match\_results, 1491  
     begin, 1491  
     cbegin, 1491  
     cend, 1492  
     empty, 1492  
     end, 1492  
     format, 1492, 1493  
     get\_allocator, 1493  
     length, 1493  
     match\_results, 1491  
     max\_size, 1493  
     operator=, 1493, 1494  
     operator[], 1494  
     position, 1494  
     prefix, 1494  
     ready, 1495  
     size, 1495  
     str, 1495  
     suffix, 1495  
     swap, 1496  
 std::match\_results< \_Bi\_iter, \_Alloc >, 1487  
 std::mem\_fun1\_ref\_t  
     first\_argument\_type, 1497  
     result\_type, 1497  
     second\_argument\_type, 1497  
 std::mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >, 1496  
 std::mem\_fun1\_t  
     first\_argument\_type, 1498  
     result\_type, 1498  
     second\_argument\_type, 1498  
 std::mem\_fun1\_t< \_Ret, \_Tp, \_Arg >, 1498  
 std::mem\_fun\_ref\_t  
     argument\_type, 1499  
     result\_type, 1500  
 std::mem\_fun\_ref\_t< \_Ret, \_Tp >, 1499  
 std::mem\_fun\_t  
     argument\_type, 1501  
     result\_type, 1501  
 std::mem\_fun\_t< \_Ret, \_Tp >, 1500  
 std::mersenne\_twister\_engine  
     discard, 1503  
     max, 1503  
     mersenne\_twister\_engine, 1503  
     min, 1503  
     operator<<, 1504  
     operator>>, 1504  
     operator==, 1504  
     result\_type, 1503  
 std::mersenne\_twister\_engine< \_UIntType, \_\_w, \_\_n, \_\_←  
     \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l,  
     \_\_f >, 1501  
 std::messages  
     ~messages, 1507  
     char\_type, 1507  
     id, 1507  
     messages, 1507  
     string\_type, 1507

`std::messages< _CharT >`, 1505  
`std::messages_base`, 1508  
`std::messages_byname`  
    `id`, 1510  
`std::messages_byname< _CharT >`, 1509  
`std::minus`  
    `first_argument_type`, 1511  
    `result_type`, 1511  
    `second_argument_type`, 1511  
`std::minus< _Tp >`, 1510  
`std::modulus`  
    `first_argument_type`, 1512  
    `result_type`, 1512  
    `second_argument_type`, 1512  
`std::modulus< _Tp >`, 1512  
`std::money_base`, 1513  
`std::money_get`  
    `~money_get`, 1516  
    `char_type`, 1515  
    `do_get`, 1516  
    `get`, 1516, 1517  
    `id`, 1518  
    `iter_type`, 1515  
    `money_get`, 1516  
    `string_type`, 1515  
`std::money_get< _CharT, _InIter >`, 1514  
`std::money_put`  
    `~money_put`, 1520  
    `char_type`, 1519  
    `do_put`, 1520  
    `id`, 1523  
    `iter_type`, 1519  
    `money_put`, 1520  
    `put`, 1521  
    `string_type`, 1519  
`std::money_put< _CharT, _OutIter >`, 1518  
`std::moneypunct`  
    `~moneypunct`, 1526  
    `char_type`, 1525  
    `curr_symbol`, 1526  
    `decimal_point`, 1526  
    `do_curr_symbol`, 1527  
    `do_decimal_point`, 1527  
    `do_frac_digits`, 1527  
    `do_grouping`, 1527  
    `do_neg_format`, 1528  
    `do_negative_sign`, 1528  
    `do_pos_format`, 1528  
    `do_positive_sign`, 1529  
    `do_thousands_sep`, 1529  
    `frac_digits`, 1529  
    `grouping`, 1530  
    `id`, 1532  
    `intl`, 1532  
    `moneypunct`, 1525, 1526  
    `neg_format`, 1530  
    `negative_sign`, 1531  
    `pos_format`, 1531  
    `positive_sign`, 1531  
    `string_type`, 1525  
    `thousands_sep`, 1532  
`std::moneypunct< _CharT, _Intl >`, 1523  
`std::moneypunct_byname`  
    `curr_symbol`, 1534  
    `decimal_point`, 1535  
    `do_curr_symbol`, 1535  
    `do_decimal_point`, 1535  
    `do_frac_digits`, 1535  
    `do_grouping`, 1536  
    `do_neg_format`, 1536  
    `do_negative_sign`, 1536  
    `do_pos_format`, 1537  
    `do_positive_sign`, 1537  
    `do_thousands_sep`, 1537  
    `frac_digits`, 1538  
    `grouping`, 1538  
    `id`, 1540  
    `neg_format`, 1538  
    `negative_sign`, 1539  
    `pos_format`, 1539  
    `positive_sign`, 1540  
    `thousands_sep`, 1540  
`std::moneypunct_byname< _CharT, _Intl >`, 1533  
`std::move_iterator< _Iterator >`, 1541  
`std::multimap`  
    `begin`, 1546  
    `cbegin`, 1547  
    `cend`, 1547  
    `clear`, 1547  
    `count`, 1547  
    `crbegin`, 1547  
    `crend`, 1548  
    `emplace`, 1548  
    `emplace_hint`, 1548  
    `empty`, 1548  
    `end`, 1549  
    `equal_range`, 1549  
    `erase`, 1550  
    `find`, 1552  
    `get_allocator`, 1552  
    `insert`, 1552, 1554  
    `key_comp`, 1556  
    `lower_bound`, 1556  
    `max_size`, 1556  
    `multimap`, 1544, 1546  
    `operator=`, 1557  
    `rbegin`, 1557  
    `rend`, 1558

- size, 1558
- swap, 1558
- upper\_bound, 1558, 1559
- value\_comp, 1559
- std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 1542
- std::multiplies
  - first\_argument\_type, 1560
  - result\_type, 1560
  - second\_argument\_type, 1560
- std::multiplies< \_Tp >, 1559
- std::multiset
  - begin, 1564
  - cbegin, 1564
  - cend, 1564
  - clear, 1564
  - count, 1564
  - crbegin, 1566
  - crend, 1566
  - emplace, 1566
  - emplace\_hint, 1566
  - empty, 1567
  - end, 1567
  - equal\_range, 1567
  - erase, 1568
  - find, 1570
  - get\_allocator, 1570
  - insert, 1570, 1572
  - key\_comp, 1574
  - lower\_bound, 1574
  - max\_size, 1574
  - multiset, 1562–1564
  - operator=, 1574, 1576
  - rbegin, 1576
  - rend, 1576
  - size, 1576
  - swap, 1576
  - upper\_bound, 1578
  - value\_comp, 1578
- std::multiset< \_Key, \_Compare, \_Alloc >, 1560
- std::negate
  - argument\_type, 1579
  - result\_type, 1579
- std::negate< \_Tp >, 1579
- std::negative\_binomial\_distribution
  - k, 1581
  - max, 1581
  - min, 1581
  - operator<<, 1582
  - operator>>, 1582
  - operator(), 1581
  - operator==, 1582
  - p, 1581
  - param, 1581
  - reset, 1582
  - result\_type, 1581
- std::negative\_binomial\_distribution< \_IntType >, 1580
- std::negative\_binomial\_distribution< \_IntType >::param←\_type, 1583
- std::nested\_exception, 1583
- std::normal\_distribution
  - max, 1585
  - mean, 1585
  - min, 1585
  - normal\_distribution, 1585
  - operator<<, 1586
  - operator>>, 1586
  - operator(), 1585
  - operator==, 1586
  - param, 1585
  - reset, 1586
  - result\_type, 1585
  - stddev, 1586
- std::normal\_distribution< \_RealType >, 1584
- std::normal\_distribution< \_RealType >::param\_type, 1587
- std::not\_equal\_to
  - first\_argument\_type, 1588
  - result\_type, 1588
  - second\_argument\_type, 1588
- std::not\_equal\_to< \_Tp >, 1588
- std::num\_get
  - ~num\_get, 1591
  - char\_type, 1591
  - do\_get, 1591–1596
  - get, 1597–1602
  - id, 1603
  - iter\_type, 1591
  - num\_get, 1591
- std::num\_get< \_CharT, \_InIter >, 1589
- std::num\_put
  - ~num\_put, 1605
  - char\_type, 1605
  - do\_put, 1605, 1607–1609
  - id, 1614
  - iter\_type, 1605
  - num\_put, 1605
  - put, 1609–1614
- std::num\_put< \_CharT, \_OutIter >, 1603
- std::numpunct
  - ~numpunct, 1618
  - char\_type, 1616
  - decimal\_point, 1618
  - do\_decimal\_point, 1618
  - do\_falsename, 1619
  - do\_grouping, 1619
  - do\_thousands\_sep, 1619
  - do\_truename, 1619
  - falsename, 1620

- grouping, 1620
- id, 1621
- numpunct, 1617, 1618
- string\_type, 1616
- thousands\_sep, 1620
- truename, 1621
- std::numpunct< \_CharT >, 1615
- std::numpunct\_byname
  - decimal\_point, 1623
  - do\_decimal\_point, 1623
  - do\_falsename, 1624
  - do\_grouping, 1624
  - do\_thousands\_sep, 1624
  - do\_truename, 1624
  - falsename, 1625
  - grouping, 1625
  - id, 1626
  - thousands\_sep, 1625
  - truename, 1626
- std::numpunct\_byname< \_CharT >, 1622
- std::ostream\_iterator
  - char\_type, 1628
  - difference\_type, 1628
  - iterator\_category, 1628
  - operator=, 1629
  - ostream\_iterator, 1629
  - ostream\_type, 1628
  - pointer, 1628
  - reference, 1628
  - traits\_type, 1628
  - value\_type, 1628
- std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, 1627
- std::ostreambuf\_iterator
  - char\_type, 1631
  - difference\_type, 1631
  - failed, 1632
  - iterator\_category, 1631
  - operator\*, 1632
  - operator++, 1632
  - operator=, 1632
  - ostream\_type, 1631
  - ostreambuf\_iterator, 1632
  - pointer, 1631
  - reference, 1631
  - streambuf\_type, 1631
  - traits\_type, 1631
  - value\_type, 1632
- std::ostreambuf\_iterator< \_CharT, \_Traits >, 1630
- std::output\_iterator\_tag, 1633
- std::owner\_less< \_Tp >, 1633
- std::owner\_less< shared\_ptr< \_Tp > >, 1633
  - first\_argument\_type, 1634
  - result\_type, 1634
  - second\_argument\_type, 1634
- std::owner\_less< weak\_ptr< \_Tp > >, 1634
  - first\_argument\_type, 1635
  - result\_type, 1635
  - second\_argument\_type, 1635
- std::pair
  - first, 1637
  - pair, 1636, 1637
  - second, 1637
  - second\_type, 1636
- std::pair< \_T1, \_T2 >, 1635
- std::piecewise\_constant\_distribution
  - densities, 1638
  - intervals, 1639
  - max, 1639
  - min, 1639
  - operator<<, 1640
  - operator>>, 1640
  - operator(), 1639
  - operator==, 1640
  - param, 1639
  - reset, 1639
  - result\_type, 1638
- std::piecewise\_constant\_distribution< \_RealType >, 1637
- std::piecewise\_constant\_distribution< \_RealType >←
  - ::param\_type, 1640
- std::piecewise\_construct\_t, 1641
- std::piecewise\_linear\_distribution
  - densities, 1643
  - intervals, 1643
  - max, 1643
  - min, 1643
  - operator<<, 1644
  - operator>>, 1644
  - operator(), 1643
  - operator==, 1644
  - param, 1643
  - reset, 1644
  - result\_type, 1642
- std::piecewise\_linear\_distribution< \_RealType >, 1641
- std::piecewise\_linear\_distribution< \_RealType >←
  - ::param\_type, 1645
- std::plus
  - first\_argument\_type, 1646
  - result\_type, 1646
  - second\_argument\_type, 1646
- std::plus< \_Tp >, 1645
- std::pointer\_to\_binary\_function
  - first\_argument\_type, 1647
  - result\_type, 1647
  - second\_argument\_type, 1648
- std::pointer\_to\_binary\_function< \_Arg1, \_Arg2, \_Result >, 1647
- std::pointer\_to\_unary\_function
  - argument\_type, 1649

- result\_type, 1649
- std::pointer\_to\_unary\_function< \_Arg, \_Result >, 1648
- std::pointer\_traits
  - difference\_type, 1650
  - element\_type, 1650
  - pointer, 1650
- std::pointer\_traits< \_Ptr >, 1649
- std::pointer\_traits< \_Tp \* >, 1650
  - difference\_type, 1650
  - element\_type, 1650
  - pointer, 1651
  - pointer\_to, 1651
- std::poisson\_distribution
  - max, 1652
  - mean, 1652
  - min, 1652
  - operator<<, 1653
  - operator>>, 1654
  - operator(), 1653
  - operator==, 1654
  - param, 1653
  - reset, 1653
  - result\_type, 1652
- std::poisson\_distribution< \_IntType >, 1651
- std::poisson\_distribution< \_IntType >::param\_type, 1654
- std::priority\_queue
  - empty, 1657
  - pop, 1657
  - priority\_queue, 1656
  - push, 1657
  - size, 1657
  - top, 1657
- std::priority\_queue< \_Tp, \_Sequence, \_Compare >, 1655
- std::queue
  - back, 1659
  - c, 1660
  - empty, 1659
  - front, 1659
  - pop, 1660
  - push, 1660
  - queue, 1659
  - size, 1660
- std::queue< \_Tp, \_Sequence >, 1658
- std::random\_access\_iterator\_tag, 1661
- std::random\_device
  - result\_type, 1662
- std::raw\_storage\_iterator
  - difference\_type, 1663
  - iterator\_category, 1663
  - pointer, 1663
  - reference, 1663
  - value\_type, 1663
- std::raw\_storage\_iterator< \_OutputIterator, \_Tp >, 1662
- std::regex\_constants, 429
- std::regex\_error, 1664
  - code, 1664
  - regex\_error, 1664
- std::regex\_iterator
  - operator!=, 1666
  - operator\*, 1666
  - operator++, 1666
  - operator->, 1666
  - operator=, 1666
  - operator==, 1667
  - regex\_iterator, 1665
- std::regex\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, 1664
- std::regex\_token\_iterator
  - operator!=, 1670
  - operator\*, 1670
  - operator++, 1671
  - operator->, 1671
  - operator=, 1671
  - operator==, 1671
  - regex\_token\_iterator, 1668, 1670
- std::regex\_token\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, 1667
- std::regex\_traits
  - getloc, 1673
  - imbue, 1673
  - length, 1673
  - lookup\_classname, 1673
  - lookup\_collatename, 1674
  - regex\_traits, 1672
  - transform, 1674
  - transform\_primary, 1676
  - translate, 1676
  - translate\_nocase, 1676
- std::regex\_traits< \_Ch\_type >, 1672
- std::rel\_ops, 430
  - operator!=, 430
  - operator<=, 431
  - operator>, 431
  - operator>=, 431
- std::reverse\_iterator
  - base, 1679
  - iterator\_category, 1678
  - operator\*, 1679
  - operator+, 1680
  - operator++, 1680
  - operator+=, 1680
  - operator-, 1680
  - operator->, 1681
  - operator--, 1681
  - operator=, 1681
  - operator[], 1681
  - reverse\_iterator, 1679
  - value\_type, 1678

- std::reverse\_iterator< \_Iterator >, 1677
- std::seed\_seq, 1682
  - result\_type, 1682
  - seed\_seq, 1682
- std::set
  - allocator\_type, 1685
  - begin, 1689
  - cbegin, 1689
  - cend, 1689
  - clear, 1689
  - const\_iterator, 1685
  - const\_pointer, 1685
  - const\_reference, 1685
  - const\_reverse\_iterator, 1685
  - count, 1689
  - crbegin, 1691
  - crend, 1691
  - difference\_type, 1685
  - emplace, 1691
  - emplace\_hint, 1691
  - empty, 1692
  - end, 1692
  - equal\_range, 1692
  - erase, 1694
  - find, 1696
  - get\_allocator, 1696
  - insert, 1696, 1698
  - iterator, 1685
  - key\_comp, 1700
  - key\_compare, 1685
  - key\_type, 1685
  - lower\_bound, 1700
  - max\_size, 1700
  - operator=, 1700, 1701
  - pointer, 1686
  - rbegin, 1701
  - reference, 1686
  - rend, 1701
  - reverse\_iterator, 1686
  - set, 1686, 1688, 1689
  - size, 1701
  - size\_type, 1686
  - swap, 1701
  - upper\_bound, 1702
  - value\_comp, 1702
  - value\_compare, 1686
  - value\_type, 1686
- std::set< \_Key, \_Compare, \_Alloc >, 1683
- std::shared\_ptr
  - allocate\_shared, 1707
  - shared\_ptr, 1704–1707
- std::shared\_ptr< \_Tp >, 1702
- std::shuffle\_order\_engine
  - base, 1712
  - discard, 1712
  - max, 1712
  - min, 1712
  - operator<<, 1713
  - operator>>, 1714
  - operator(), 1713
  - operator==, 1713
  - result\_type, 1710
  - seed, 1713
  - shuffle\_order\_engine, 1710, 1712
- std::shuffle\_order\_engine< \_RandomNumberEngine, \_↔\_k >, 1709
- std::slice, 1714
- std::slice\_array< \_Tp >, 1715
- std::stack
  - empty, 1717
  - pop, 1717
  - push, 1717
  - size, 1718
  - stack, 1717
  - top, 1718
- std::stack< \_Tp, \_Sequence >, 1716
- std::student\_t\_distribution
  - max, 1719
  - min, 1719
  - operator<<, 1720
  - operator>>, 1721
  - operator(), 1720
  - operator==, 1720
  - param, 1720
  - reset, 1720
  - result\_type, 1719
- std::student\_t\_distribution< \_RealType >, 1718
- std::student\_t\_distribution< \_RealType >::param\_type, 1721
- std::sub\_match
  - compare, 1723
  - first, 1724
  - length, 1724
  - operator string\_type, 1724
  - second, 1724
  - second\_type, 1723
  - str, 1724
- std::sub\_match< \_Bilter >, 1722
- std::time\_base, 1725
- std::time\_get
  - ~time\_get, 1728
  - char\_type, 1727
  - date\_order, 1728
  - do\_date\_order, 1728
  - do\_get\_date, 1729
  - do\_get\_monthname, 1729
  - do\_get\_time, 1730
  - do\_get\_weekday, 1730

- do\_get\_year, 1731
- get\_date, 1731
- get\_monthname, 1732
- get\_time, 1732
- get\_weekday, 1733
- get\_year, 1733
- id, 1734
- iter\_type, 1727
- time\_get, 1728
- std::time\_get< \_CharT, \_InIter >, 1726
- std::time\_get\_byname
  - date\_order, 1736
  - do\_date\_order, 1736
  - do\_get\_date, 1736
  - do\_get\_monthname, 1737
  - do\_get\_time, 1737
  - do\_get\_weekday, 1738
  - do\_get\_year, 1738
  - get\_date, 1739
  - get\_monthname, 1739
  - get\_time, 1740
  - get\_weekday, 1740
  - get\_year, 1741
  - id, 1741
- std::time\_get\_byname< \_CharT, \_InIter >, 1734
- std::time\_put
  - ~time\_put, 1743
  - char\_type, 1743
  - do\_put, 1744
  - id, 1745
  - iter\_type, 1743
  - put, 1744
  - time\_put, 1743
- std::time\_put< \_CharT, \_OutIter >, 1742
- std::time\_put\_byname
  - do\_put, 1746
  - id, 1748
  - put, 1747
- std::time\_put\_byname< \_CharT, \_OutIter >, 1745
- std::tr1, 432
- std::tr1::\_\_detail, 432
- std::tr2, 432
- std::tr2::\_\_detail, 432
- std::unary\_function
  - argument\_type, 1749
  - result\_type, 1749
- std::unary\_function< \_Arg, \_Result >, 1748
- std::unary\_negate
  - argument\_type, 1750
  - result\_type, 1750
- std::unary\_negate< \_Predicate >, 1749
- std::uniform\_int\_distribution
  - max, 1751
  - min, 1752
- operator(), 1752
- operator==, 1752
- param, 1752
- reset, 1752
- result\_type, 1751
- uniform\_int\_distribution, 1751
- std::uniform\_int\_distribution< \_IntType >, 1750
- std::uniform\_int\_distribution< \_IntType >::param\_type, 1753
- std::uniform\_real\_distribution
  - max, 1754
  - min, 1755
  - operator(), 1755
  - operator==, 1755
  - param, 1755
  - reset, 1755
  - result\_type, 1754
  - uniform\_real\_distribution, 1754
- std::uniform\_real\_distribution< \_RealType >, 1753
- std::uniform\_real\_distribution< \_RealType >::param\_type, 1756
- std::unique\_ptr< \_Tp, \_Dp >, 1756
- std::unique\_ptr< \_Tp[], \_Dp >, 1757
- std::unordered\_map
  - allocator\_type, 1761
  - at, 1764, 1765
  - begin, 1765, 1766
  - bucket\_count, 1766
  - cbegin, 1766
  - cend, 1766, 1767
  - clear, 1767
  - const\_iterator, 1761
  - const\_local\_iterator, 1761
  - const\_pointer, 1761
  - const\_reference, 1761
  - count, 1767
  - difference\_type, 1761
  - emplace, 1767
  - emplace\_hint, 1768
  - empty, 1768
  - end, 1768, 1769
  - equal\_range, 1769
  - erase, 1771, 1772
  - find, 1772
  - get\_allocator, 1773
  - hash\_function, 1773
  - hasher, 1762
  - insert, 1773–1775
  - iterator, 1762
  - key\_eq, 1775
  - key\_equal, 1762
  - key\_type, 1762
  - load\_factor, 1775
  - local\_iterator, 1762

- mapped\_type, 1762
- max\_bucket\_count, 1775
- max\_load\_factor, 1776
- max\_size, 1776
- operator=, 1776
- operator[], 1777
- pointer, 1762
- reference, 1763
- rehash, 1777
- reserve, 1778
- size, 1778
- size\_type, 1763
- swap, 1778
- unordered\_map, 1763, 1764
- value\_type, 1763
- std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 1758
- std::unordered\_multimap
  - allocator\_type, 1781
  - begin, 1784, 1785
  - bucket\_count, 1785
  - cbegin, 1785
  - cend, 1787
  - clear, 1787
  - const\_iterator, 1781
  - const\_local\_iterator, 1781
  - const\_pointer, 1781
  - const\_reference, 1781
  - count, 1787
  - difference\_type, 1781
  - emplace, 1787
  - emplace\_hint, 1789
  - empty, 1789
  - end, 1789, 1790
  - equal\_range, 1790
  - erase, 1792, 1793
  - find, 1793
  - get\_allocator, 1794
  - hash\_function, 1794
  - hasher, 1782
  - insert, 1794–1796
  - iterator, 1782
  - key\_eq, 1796
  - key\_equal, 1782
  - key\_type, 1782
  - load\_factor, 1796
  - local\_iterator, 1782
  - mapped\_type, 1782
  - max\_bucket\_count, 1796
  - max\_load\_factor, 1796
  - max\_size, 1798
  - operator=, 1798
  - pointer, 1782
  - reference, 1783
  - rehash, 1798
  - reserve, 1798
  - size, 1800
  - size\_type, 1783
  - swap, 1800
  - unordered\_multimap, 1783, 1784
  - value\_type, 1783
- std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 1778
- std::unordered\_multiset
  - allocator\_type, 1802
  - begin, 1807
  - bucket\_count, 1807
  - cbegin, 1808
  - cend, 1808
  - clear, 1809
  - const\_iterator, 1802
  - const\_local\_iterator, 1803
  - const\_pointer, 1803
  - const\_reference, 1803
  - count, 1809
  - difference\_type, 1803
  - emplace, 1809
  - emplace\_hint, 1809
  - empty, 1810
  - end, 1810
  - equal\_range, 1812
  - erase, 1812, 1813
  - find, 1814
  - get\_allocator, 1814
  - hash\_function, 1814
  - hasher, 1803
  - insert, 1815, 1816
  - iterator, 1803
  - key\_eq, 1818
  - key\_equal, 1803
  - key\_type, 1804
  - load\_factor, 1818
  - local\_iterator, 1804
  - max\_bucket\_count, 1818
  - max\_load\_factor, 1818
  - max\_size, 1818
  - operator=, 1818, 1819
  - pointer, 1804
  - reference, 1804
  - rehash, 1819
  - reserve, 1819
  - size, 1819
  - size\_type, 1804
  - swap, 1820
  - unordered\_multiset, 1804, 1806
  - value\_type, 1804
- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 1800



- std::unordered\_set
  - allocator\_type, 1822
  - begin, 1825, 1826
  - bucket\_count, 1826
  - cbegin, 1826, 1827
  - cend, 1827
  - clear, 1827
  - const\_iterator, 1822
  - const\_local\_iterator, 1822
  - const\_pointer, 1823
  - const\_reference, 1823
  - count, 1827
  - difference\_type, 1823
  - emplace, 1829
  - emplace\_hint, 1829
  - empty, 1829
  - end, 1830
  - equal\_range, 1831, 1832
  - erase, 1832, 1833
  - find, 1833, 1835
  - get\_allocator, 1835
  - hash\_function, 1835
  - hasher, 1823
  - insert, 1835–1837
  - iterator, 1823
  - key\_eq, 1838
  - key\_equal, 1823
  - key\_type, 1823
  - load\_factor, 1838
  - local\_iterator, 1823
  - max\_bucket\_count, 1838
  - max\_load\_factor, 1838
  - max\_size, 1838
  - operator=, 1838, 1839
  - pointer, 1824
  - reference, 1824
  - rehash, 1839
  - reserve, 1839
  - size, 1839
  - size\_type, 1824
  - swap, 1840
  - unordered\_set, 1824, 1825
  - value\_type, 1824
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 1820
- std::uses\_allocator< \_Tp, \_Alloc >, 1840
- std::vector
  - \_M\_allocate\_and\_copy, 1846
  - \_M\_range\_check, 1846
  - ~vector, 1846
  - assign, 1846, 1847
  - at, 1847, 1848
  - back, 1848
  - begin, 1848
  - capacity, 1849
  - cbegin, 1849
  - cend, 1849
  - clear, 1849
  - crbegin, 1849
  - crend, 1849
  - data, 1849
  - emplace, 1850
  - empty, 1850
  - end, 1850
  - erase, 1850, 1852
  - front, 1852
  - insert, 1852–1854
  - max\_size, 1854
  - operator=, 1854
  - operator[], 1856
  - pop\_back, 1856
  - push\_back, 1856
  - rbegin, 1857
  - rend, 1857
  - reserve, 1857
  - resize, 1858
  - shrink\_to\_fit, 1858
  - size, 1858
  - swap, 1858
  - vector, 1844–1846
- std::vector< \_Tp, \_Alloc >, 1841
- std::vector< bool, \_Alloc >, 1859
- std::weak\_ptr< \_Tp >, 1862
- std::weibull\_distribution
  - a, 1864
  - b, 1864
  - max, 1864
  - min, 1864
  - operator(), 1864
  - operator==, 1865
  - param, 1864
  - reset, 1865
  - result\_type, 1864
- std::weibull\_distribution< \_RealType >, 1863
- std::weibull\_distribution< \_RealType >::param\_type, 1865
- stdc++.h, 2043
- stddev
  - std::normal\_distribution, 1586
- stdio\_filebuf
  - \_\_gnu\_cxx::stdio\_filebuf, 534
- stdio\_filebuf.h, 2043
- stdio\_sync\_filebuf.h, 2043
- stdtr1c++.h, 2044
- stl\_algo.h, 2044
- stl\_algobase.h, 2052
- stl\_bvector.h, 2054
- stl\_construct.h, 2054

- stl\_deque.h, 2055
- \_GLIBCXX\_DEQUE\_BUF\_SIZE, 2056
- stl\_function.h, 2057
- stl\_heap.h, 2058
- stl\_iterator.h, 2059
- stl\_iterator\_base\_funcs.h, 2062
- stl\_iterator\_base\_types.h, 2063
- stl\_list.h, 2063
- stl\_map.h, 2064
- stl\_multimap.h, 2065
- stl\_multiset.h, 2065
- stl\_numeric.h, 2066
- stl\_pair.h, 2067
- stl\_queue.h, 2067
- stl\_raw\_storage\_iter.h, 2068
- stl\_relops.h, 2068
- stl\_set.h, 2069
- stl\_stack.h, 2070
- stl\_tempbuf.h, 2070
- stl\_tree.h, 2071
- stl\_uninitialized.h, 2072
- stl\_vector.h, 2073
- str
  - std::match\_results, 1495
  - std::sub\_match, 1724
- stream\_iterator.h, 2074
- streambuf\_iterator.h, 2074
- streambuf\_type
  - std::istreambuf\_iterator, 1423
  - std::ostreambuf\_iterator, 1631
- streamoff
  - std, 353
- streampos
  - std, 353
- streamsize
  - std, 354
- stride
  - Numeric\_arrays, 122
- string
  - Strings, 210
- string\_conversions.h, 2075
- string\_type
  - std::collate, 1223
  - std::collate\_byname, 1229
  - std::messages, 1507
  - std::money\_get, 1515
  - std::money\_put, 1519
  - std::moneypunct, 1525
  - std::numpunct, 1616
- stringfwd.h, 2075
- Strings, 210
  - string, 210
  - u16string, 210
  - u32string, 210
- wstring, 210
- substr
  - \_\_gnu\_cxx::\_\_versa\_string, 507
  - std::basic\_string, 1175
- suffix
  - std::match\_results, 1495
- swap
  - \_\_gnu\_cxx, 238
  - \_\_gnu\_cxx::\_\_versa\_string, 508
  - \_\_gnu\_pbds::sample\_probe\_fn, 902
  - \_\_gnu\_pbds::sample\_range\_hashing, 903
  - \_\_gnu\_pbds::sample\_ranged\_hash\_fn, 904
  - \_\_gnu\_pbds::sample\_resize\_policy, 907
  - \_\_gnu\_pbds::sample\_resize\_trigger, 909
  - \_\_gnu\_pbds::sample\_size\_policy, 910
  - \_\_gnu\_pbds::sample\_update\_policy, 913
- Regular Expressions, 174
- std, 405, 406
  - std::\_\_detail::\_\_Nfa, 1045
  - std::basic\_regex, 1130
  - std::basic\_string, 1176
  - std::deque, 1323
  - std::forward\_list, 1362
  - std::list, 1451
  - std::map, 1483
  - std::match\_results, 1496
  - std::multimap, 1558
  - std::multiset, 1576
  - std::set, 1701
  - std::unordered\_map, 1778
  - std::unordered\_multimap, 1800
  - std::unordered\_multiset, 1820
  - std::unordered\_set, 1840
  - std::vector, 1858
- Utilities, 221, 222
- swap\_ranges
  - Mutating, 88
- sync\_with\_stdio
  - std::basic\_ios, 1117
  - std::ios\_base, 1414
- syntax\_option\_type
  - Regular Expressions, 145
- synth\_access\_traits
  - \_\_gnu\_pbds::detail::trie\_traits< Key, Mapped, \_↵  
 ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >, 863
  - \_\_gnu\_pbds::detail::trie\_traits< Key, null\_type, \_↵  
 ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >, 864
- synth\_access\_traits.hpp, 2076
- t
  - std::binomial\_distribution, 1189
- TLB\_size

- `__gnu_parallel::Settings`, 674
- table
  - `std::ctype< char >`, 1257
  - `std::ctype_byname< char >`, 1297
- table\_size
  - `std::ctype< char >`, 1260
  - `std::ctype_byname< char >`, 1299
- `tag_and_trait.hpp`, 2076
- Tags, 211
  - `trivial_iterator_difference_type`, 211
- `tags.h`, 2077
- `tgmath.h`, 2078
- `thin_heap_.hpp`, 2078
- thousands\_sep
  - `std::moneypunct`, 1532
  - `std::moneypunct_byname`, 1540
  - `std::numpunct`, 1620
  - `std::numpunct_byname`, 1625
- `throw_allocator.h`, 2079
- `throw_with_nested`
  - Exceptions, 47
- tie
  - `std::basic_ios`, 1118
- time
  - `std::locale`, 1457
- time\_get
  - `std::time_get`, 1728
- `time_members.h`, 2080
- time\_put
  - `std::time_put`, 1743
- tolower
  - `std`, 408
  - `std::__ctype_abstract_base`, 981
  - `std::ctype`, 1246, 1247
  - `std::ctype< char >`, 1258
  - `std::ctype< wchar_t >`, 1272
  - `std::ctype_byname`, 1286, 1287
  - `std::ctype_byname< char >`, 1297
- top
  - `std::priority_queue`, 1657
  - `std::stack`, 1718
- toupper
  - `std`, 408
  - `std::__ctype_abstract_base`, 981, 982
  - `std::ctype`, 1247
  - `std::ctype< char >`, 1258, 1259
  - `std::ctype< wchar_t >`, 1274
  - `std::ctype_byname`, 1287
  - `std::ctype_byname< char >`, 1298
- `trace_fn_imps.hpp`, 2080, 2081
- Traits, 212
  - `erase_can_throw`, 213
  - `order_preserving`, 213
  - `reverse_iteration`, 213

- `split_join_can_throw`, 213
- `traits.hpp`, 2081–2083
- `traits_type`
  - `std::basic_ios`, 1109
  - `std::istreambuf_iterator`, 1423
  - `std::ostream_iterator`, 1628
  - `std::ostreambuf_iterator`, 1631
- transform
  - Mutating, 88, 89
  - `std::collate`, 1227
  - `std::collate_byname`, 1231
  - `std::regex_traits`, 1674
- transform\_minimal\_n
  - `__gnu_parallel::Settings`, 674
- transform\_primary
  - `std::regex_traits`, 1676
- translate
  - `std::regex_traits`, 1676
- translate\_nocase
  - `std::regex_traits`, 1676
- tree
  - `__gnu_pbds::tree`, 918, 919
- `tree_policy.hpp`, 2083
- `tree_trace_base.hpp`, 2084
- trie
  - `__gnu_pbds::trie`, 923, 924
- `trie_policy.hpp`, 2084
- `trie_policy_base.hpp`, 2084
- `trie_string_access_traits_imp.hpp`, 2085
- `trivial_iterator_difference_type`
  - Tags, 211
- trunename
  - `std::numpunct`, 1621
  - `std::numpunct_byname`, 1626
- trunc
  - `std::basic_ios`, 1123
  - `std::ios_base`, 1418
- type
  - `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_TI >`, 756
  - `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_TI >`, 757
  - `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_TI >`, 758
  - `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_TI >`, 758
  - `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_TI >`, 759
  - `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_TI >`, 760
  - `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_TI >`, 760

- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, gp\_hash\_tag, Policy\_Tl >, 761
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, list\_update\_tag, Policy\_Tl >, 761
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, ov\_tree\_tag, Policy\_Tl >, 762
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, pat\_trie\_tag, Policy\_Tl >, 763
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, splay\_tree\_tag, Policy\_Tl >, 763
- \_\_gnu\_pbds::detail::container\_base\_dispatch< \_V< Tp, Cmp\_Fn, \_Alloc, binary\_heap\_tag, null\_type >, 754
- \_\_gnu\_pbds::detail::container\_base\_dispatch< \_V< Tp, Cmp\_Fn, \_Alloc, binomial\_heap\_tag, null\_type >, 754
- \_\_gnu\_pbds::detail::container\_base\_dispatch< \_V< Tp, Cmp\_Fn, \_Alloc, pairing\_heap\_tag, null\_type >, 755
- \_\_gnu\_pbds::detail::container\_base\_dispatch< \_V< VTp, Cmp\_Fn, \_Alloc, rc\_binomial\_heap\_tag, null\_type >, 755
- \_\_gnu\_pbds::detail::container\_base\_dispatch< \_V< Tp, Cmp\_Fn, \_Alloc, thin\_heap\_tag, null\_type >, 756
- \_\_gnu\_pbds::detail::default\_comb\_hash\_fn, 764
- \_\_gnu\_pbds::detail::default\_eq\_fn, 765
- \_\_gnu\_pbds::detail::default\_hash\_fn, 765
- \_\_gnu\_pbds::detail::default\_probe\_fn, 766
- \_\_gnu\_pbds::detail::default\_resize\_policy, 766
- \_\_gnu\_pbds::detail::default\_trie\_access\_traits< std::basic\_string< Char, Char\_Traits, std::allocator< char > >, 767
- \_\_gnu\_pbds::detail::default\_update\_policy, 767
- \_\_gnu\_pbds::detail::entry\_cmp< \_VTp, Cmp\_Fn, \_Alloc, true >, 769
- type\_traits.h, 2085
- type\_utils.hpp, 2085
- typelist.h, 2086
- types.h, 2087
- types\_traits.hpp, 2087
- u16streampos
  - std, 354
- u16string
  - Strings, 210
- u32streampos
  - std, 354
- u32string
  - Strings, 210
- Uniform Distributions, 214
  - operator!=, 214
  - operator<<, 214, 216
  - operator>>, 216
  - uniform\_int\_distribution
    - std::uniform\_int\_distribution, 1751
  - uniform\_real\_distribution
    - std::uniform\_real\_distribution, 1754
  - uninitialized\_copy
    - std, 408
  - uninitialized\_copy\_n
    - std, 408
  - uninitialized\_fill
    - std, 409
  - uninitialized\_fill\_n
    - std, 409
  - unique
    - Mutating, 89
    - std::forward\_list, 1363
    - std::list, 1451
  - unique\_copy
    - Mutating, 91
  - unique\_copy.h, 2088
  - unique\_copy\_minimal\_n
    - \_\_gnu\_parallel::Settings, 674
  - unique\_ptr.h, 2088
  - unitbuf
    - std, 409
    - std::basic\_ios, 1123
    - std::ios\_base, 1418
  - Unordered Associative, 217
  - unordered\_base.h, 2089
  - unordered\_map
    - std::unordered\_map, 1763, 1764
  - unordered\_map.h, 2090
  - unordered\_multimap
    - std::unordered\_multimap, 1783, 1784
  - unordered\_multiset
    - std::unordered\_multiset, 1804, 1806
  - unordered\_set
    - std::unordered\_set, 1824, 1825
  - unordered\_set.h, 2091
  - unsetf
    - std::basic\_ios, 1118
    - std::ios\_base, 1414
  - unshift
    - std::\_\_codecvt\_abstract\_base, 971
    - std::codecvt, 1204
    - std::codecvt< \_InternT, \_ExternT, encoding\_state >, 1208
    - std::codecvt< char, char, mbstate\_t >, 1212
    - std::codecvt< wchar\_t, char, mbstate\_t >, 1216
    - std::codecvt\_byname, 1221
  - update\_fn\_imps.hpp, 2091
  - upper\_bound

- Binary Search, 33
- std::map, 1485
- std::multimap, 1558, 1559
- std::multiset, 1578
- std::set, 1702
- uppercase
  - std, 409
  - std::basic\_ios, 1123
  - std::ios\_base, 1418
- Utilities, 218
  - \_\_addressof, 219
  - addressof, 219
  - forward, 219
  - make\_pair, 220
  - move, 220
  - move\_if\_noexcept, 220
  - operator!=, 221
  - operator<, 221
  - operator<=, 221
  - operator>, 221
  - operator>=, 221
  - operator==, 221
  - piecewise\_construct, 222
  - swap, 221, 222
- valarray\_after.h, 2092
- valarray\_array.h, 2099
- valarray\_before.h, 2105
- valid\_prefix
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer, 818
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_iter, 821
- value
  - Regular Expressions, 174
- value\_comp
  - std::map, 1485
  - std::multimap, 1559
  - std::multiset, 1578
  - std::set, 1702
- value\_compare
  - std::set, 1686
- value\_type
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_↔  
it\_, 725
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_, 730
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_,  
737
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_↔  
iterator\_, 740
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_↔  
const\_iterator\_, 781
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_↔  
node\_point\_const\_iterator\_, 786
  - const\_iterator\_, 953
  - iterator\_, 957
  - point\_const\_iterator\_, 961
  - point\_iterator\_, 963
  - std::allocator\_traits, 1091
  - std::back\_insert\_iterator, 1101
  - std::front\_insert\_iterator, 1366
  - std::insert\_iterator, 1404
  - std::istream\_iterator, 1421
  - std::istreambuf\_iterator, 1423
  - std::iterator, 1426
  - std::ostream\_iterator, 1628
  - std::ostreambuf\_iterator, 1632
  - std::raw\_storage\_iterator, 1663
  - std::reverse\_iterator, 1678
  - std::set, 1686
  - std::unordered\_map, 1763
  - std::unordered\_multimap, 1783
  - std::unordered\_multiset, 1804
  - std::unordered\_set, 1824
- vector
  - std::vector, 1844–1846
- void\_pointer
  - \_\_gnu\_cxx::\_\_alloc\_traits, 434
  - std::allocator\_traits, 1091
- vstring.h, 2105
- vstring\_fwd.h, 2107
- vstring\_util.h, 2108
- wcregex\_token\_iterator
  - Regular Expressions, 145
- wcsub\_match
  - Regular Expressions, 146
- widen
  - std::\_\_ctype\_abstract\_base, 982
  - std::basic\_ios, 1118
  - std::ctype, 1248
  - std::ctype< char >, 1259
  - std::ctype< wchar\_t >, 1274, 1275
  - std::ctype\_byname, 1288
  - std::ctype\_byname< char >, 1298, 1299
- width
  - std::basic\_ios, 1119
  - std::ios\_base, 1414
- workstealing.h, 2108
- wregex
  - Regular Expressions, 146
- wsregex\_token\_iterator
  - Regular Expressions, 146
- wssub\_match
  - Regular Expressions, 146
- wstreampos
  - std, 354
- wstring
  - Strings, 210
- xalloc

`std::basic_ios`, [1119](#)

`std::ios_base`, [1415](#)