

APPENDIX



Standard Library Headers

The C++ Standard Library consists of 88 header files, of which 6 are deprecated, and 26 are adapted from the C Standard Library. This appendix gives a brief description of each.

For each `<name.h>` header from the C Standard Library, there is a corresponding `<cname>` C++ Standard Library header (note the ‘c’ prefix). These C++ headers put all functionality provided by the C library in the `std` namespace. It is implementation-defined whether the types and functions still appear in the global namespace. The use of the original `<name.h>` headers is deprecated.

Headers are shown in the order in which they are presented in each chapter. Some are covered by multiple chapters. Functionality not discussed in this book is shown in *italic*.

Numerics and Math (Chapter 1)

Header	Contents
<code><cmath></code>	Math functions, such as <code>exp()</code> , <code>sqrt()</code> , <code>log()</code> , <code>abs()</code> , all trigonometric functions, special mathematical functions, and more.
<code><numeric></code>	The <code>gcd()</code> and <code>lcm()</code> functions (+ several algorithms: see Chapter 4).
<code><algorithm></code>	<code>min()</code> , <code>max()</code> , <code>minmax()</code> , and <code>clamp()</code> (+ many algorithms: Chapter 4).
<code><cstdint></code>	A set of type aliases for integral types with certain width requirements, e.g., <code>int32_t</code> and <code>int_fast64_t</code> .
<code><limits></code>	<code>numeric_limits</code> , offering properties—such as <code>min()</code> , <code>max()</code> , <code>lowest()</code> , <code>infinity()</code> , <code>quiet_NaN()</code> , etc.—for all built-in arithmetic types.
<code><climits></code>	<i>Macros for C-style limits of integral types, such as <code>INT_MAX</code>. Subsumed by <code><limits></code>.</i>
<code><cmath></code>	<i>Macros to describe details of the floating-point types of your environment, e.g., <code>FLT_EPSILON</code>, <code>FLT_MAX</code>, etc. Subsumed by <code><limits></code>.</i>
<code><cfenv></code>	<i>Advanced access to the floating-point environment to configure floating-point exceptions, rounding, and other environment settings.</i>
<code><complex></code>	The <code>complex</code> class for working with complex numbers.

(continued)

Header	Contents
<code><complex></code>	Simply includes <code><complex></code> . <i>Deprecated since C++17.</i>
<code><ctgmath></code>	Includes <code><cmath></code> and <code><complex></code> . <i>Deprecated since C++17.</i>
<code><ratio></code>	The ratio template, helper templates for performing arithmetic operations and comparisons on them, and a set of predefined ratios.
<code><random></code>	Pseudorandom number generators, <code>random_device</code> , and various random number distributions.
<code><valarray></code>	<code>valarray</code> functionality for working with arrays of numeric values.

General Utilities (Chapter 2)

Header	Contents
<code><utility></code>	<code>pair</code> , <code>piecewise_construct</code> , and <code>integer_sequence</code> . Functions <code>make_pair()</code> , <code>swap()</code> , <code>exchange()</code> , <code>forward()</code> , <code>move()</code> , <code>move_if_noexcept()</code> , <code>as_const()</code> , and <code>declval()</code> .
<code><tuple></code>	<code>tuple</code> , helper classes <code>tuple_size</code> and <code>tuple_element</code> , and functions <code>make_tuple()</code> , <code>forward_as_tuple()</code> , <code>tie()</code> , <code>tuple_cat()</code> , <code>get()</code> , <code>apply()</code> , and <code>make_from_tuple()</code> .
<code><cstdint></code>	The byte type.
<code><memory></code>	Smart pointers: <code>unique_ptr</code> , <code>shared_ptr</code> , and <code>weak_ptr</code> . Cast functions: <code>static_pointer_cast()</code> , <code>dynamic_pointer_cast()</code> , <code>const_pointer_cast()</code> , and <code>reinterpret_pointer_cast()</code> . The <code>addressof()</code> function. Also: allocators (Chapter 3) and algorithms for uninitialized memory (Chapter 4).
<code><new></code>	<i>Functions for managing dynamic storage: operators <code>new</code>, <code>new[]</code>, <code>delete</code>, and <code>delete[]</code>, <code>get_and_set_new_handler()</code>, and exceptions <code>bad_alloc</code> and <code>bad_array_new_length</code>.</i>
<code><functional></code>	Reference wrappers (created with <code>ref()</code> / <code>cref()</code>), predefined functors (function objects), <code>std::bind()</code> , <code>not_fn()</code> , <code>function</code> , and <code>mem_fn()</code> . The <code>std::invoke()</code> utility to invoke any callable.
<code><initializer_list></code>	The definition of <code>initializer_list</code> .
<code><optional></code> C++17	The optional class template, <code>bad_optional_access</code> exception, <code>nullopt</code> constant, and <code>make_optional()</code> function template.
<code><variant></code> C++17	The variant class template, the <code>variant_size</code> and <code>variant_alternative</code> templates, the <code>bad_variant_access</code> exception, monostate structure, <code>variant_npos</code> constant, and the functions <code>visit()</code> , <code>holds_alternative()</code> , <code>get()</code> , and <code>get_if()</code> .

(continued)

Header	Contents
<code><any></code> <small>C++17</small>	The <code>any</code> class template, <code>bad_any_cast</code> exception, and <code>any_cast()</code> function template.
<code><chrono></code>	Time utilities: <code>durations</code> , <code>time_points</code> , and <code>clocks</code> (<code>steady_clock</code> , <code>system_clock</code> , and <code>high_resolution_clock</code>), and helper functions <code>floor()</code> , <code>ceil()</code> , <code>round()</code> , and <code>abs()</code> .
<code><ctime></code>	C-style time and date utilities such as the <code>tm</code> struct, <code>time()</code> , <code>localtime()</code> , and <code>strftime()</code> .
<code><typeinfo></code>	<code>type_info</code> , and the exceptions <code>bad_cast</code> and <code>bad_typeid</code> .
<code><typeindex></code>	<code>type_index</code> , a wrapper for <code>type_info</code> to be able to use it as a key in associative containers.
<code><type_traits></code>	Template-based type traits for compile-time manipulation and inspection of properties of types.

Containers (Chapter 3)

Header	Contents
<code><iterator></code>	Functions to perform operations on iterators: <code>advance()</code> , <code>distance()</code> , <code>begin()</code> , <code>end()</code> , <code>prev()</code> , and <code>next()</code> , and the iterator tags. See also Chapters 4 and 5.
<code><vector></code>	The <code>vector</code> class template and the <code>vector<bool></code> specialization.
<code><deque></code>	The <code>deque</code> class template.
<code><array></code>	The <code>array</code> class template.
<code><list></code>	The <code>list</code> class template.
<code><forward_list></code>	The <code>forward_list</code> class template.
<code><bitset></code>	The <code>bitset</code> class template.
<code><queue></code>	The <code>queue</code> and <code>priority_queue</code> class templates.
<code><stack></code>	The <code>stack</code> class template.
<code><map></code>	The <code>map</code> and <code>multimap</code> class templates.
<code><set></code>	The <code>set</code> and <code>multiset</code> class templates.
<code><unordered_map></code>	The <code>unordered_map</code> and <code>unordered_multimap</code> class templates.
<code><unordered_set></code>	The <code>unordered_set</code> and <code>unordered_multiset</code> class templates.
<code><memory></code>	The default allocator type <i>and allocator-related utilities</i> <code>allocator_traits</code> , <code>allocator_arg</code> , and <code>uses_allocator()</code> . Also: smart pointers (Chapter 2) and algorithms for uninitialized memory (Chapter 4).

(continued)

Header	Contents
<code><memory_resource></code> ^{C++17}	The polymorphic allocators and memory resources.
<code><scoped_allocator></code> ^{C++17}	The <code>scoped_allocator_adaptor</code> class template.

Algorithms (Chapter 4)

Header	Contents
<code><iterator></code>	Input/output iterators, and the predefined iterator adaptors: <code>reverse_iterator</code> , <code>move_iterator</code> , and insert iterators. See also Chapters 3 and 5.
<code><algorithm></code>	80 different algorithms that operate on ranges.
<code><numeric></code>	Numerical algorithms: <code>accumulate()</code> , <code>[transform_]reduce()</code> , <code>inner_product()</code> , <code>adjacent_difference()</code> , <code>partial_sum()</code> , <code>[transform_]inclusive_scan()</code> , <code>[transform_]exclusive_scan()</code> , and <code>iota()</code> .
<code><memory></code>	Algorithms for uninitialized memory: <code>destroy[_n]()</code> , <code>destroy_at()</code> , and <code>uninitialized_xxx()</code> , with <code>xxx</code> equal to <code>default_construct[_n]</code> , <code>value_construct[_n]</code> , <code>copy[_n]</code> , <code>move[_n]</code> , and <code>fill[_n]</code> . Also: smart pointers (Chapter 2) and allocators (Chapter 3).
<code><execution></code> ^{C++17}	Predefined execution policies: <code>seq</code> , <code>par</code> , and <code>par_unseq</code> .

Input/Output (Chapter 5)

Header	Contents
<code><ios></code>	<code>ios_base</code> , <code>basic_ios</code> , and <code>fpos</code> , type aliases <code>ios</code> and <code>wios</code> , and types <code>streamoff</code> , <code>streampos</code> , <code>wstreampos</code> , and <code>streamsize</code> . Nonparametric I/O manipulators such as <code>boolalpha</code> , <code>dec</code> , <code>scientific</code> , etc.
<code><iomanip></code>	Parametric I/O manipulators, such as <code>setbase()</code> , <code>setfill()</code> , <code>get_money()</code> , <code>put_time()</code> , and more.
<code><ostream></code>	<code>basic_ostream</code> , and type aliases <code>ostream</code> and <code>wostream</code> . The <code>endl</code> , <code>ends</code> , and <code>flush</code> output manipulators.
<code><istream></code>	<code>basic_istream</code> and <code>basic_iostream</code> , and type aliases <code>istream</code> , <code>wistream</code> , <code>iostream</code> , and <code>wiostream</code> . The <code>ws</code> input manipulator.

(continued)

Header	Contents
<iostream>	cin/wcin, cout/wcout, cerr/wcerr, and clog/wclog. Includes <ios>, <streambuf>, <istream>, <ostream>, and <iosfwd>.
<sstream>	String streams: basic_istringstream, basic_ostringstream, basic_stringstream, basic_stringbuf, and related type aliases.
<fstream>	File streams: basic_ifstream, basic_ofstream, basic_fstream, and basic_filebuf, and related type aliases.
<streambuf>	basic_streambuf, and type aliases streambuf and wstreambuf.
<iosfwd>	Forward declarations for all stream I/O types.
<cstdio>	The C-style I/O library. Basic file utilities remove(), rename(), tmpfile(), tmpnam(), plus fopen(), fclose(), etc. Functions for formatted (printf(), scanf(), etc.) and character-based I/O (getc(), putc(), etc.). It is generally recommended to use C++ I/O streams.
<cinttypes>	Macros to use with printf() and scanf() to handle the fixed-width integer types of <stdint>. Subsumed by C++ I/O streams.
<strstream>	Deprecated.
<iterator>	Stream iterators istream_iterator and ostream_iterator. See also Chapters 3 and 4.
<filesystem> ^{C++17}	Classes and functions to work with the file system.

Characters and Strings (Chapter 6)

Header	Contents
<string>	basic_string, and type aliases string, wstring, u16string, and u32string. Conversion functions such as stoi(), stof(), to_string(), etc.
<string_view> ^{C++17}	basic_string_view, and type aliases string_view, wstring_view, u16string_view, and u32string_view.
<cstring>	Low-level memory functions: memcpy(), memmove(), memcmp(), memchr(), and memset(). A collection of C-style string functions, e.g., strcpy() and strcat(), and a definition for NULL and size_t.
<wchar>	Functions to work with C-style wide character strings, such as fputws(), wprintf(), wcstof(), wcscat(), wmemset(), etc.
<cctype>	Functions to classify and transform characters: isdigit(), isspace(), tolower(), toupper(), etc.

(continued)

Header	Contents
<cwctype>	Wide character versions of functions from <cctype>: <code>iswdigit()</code> , <code>iswspace()</code> , <code>towlower()</code> , <code>towupper()</code> , etc.
<codecvt>	Unicode character-encoding conversion facets: <code>codecvt_utf8</code> , <code>codecvt_utf16</code> , and <code>codecvt_utf8_utf16</code> . Deprecated since C++17.
<cuchar>	<i>Functions to convert between 16- or 32-bit character and multibyte sequences: <code>c16rtomb()</code>, <code>c32rtomb()</code>, <code>mbrtoc16()</code>, <code>mbrtoc32()</code>.</i>
<locale>	The <code>locale</code> class, overloads of <cctype> functions accepting a given locale, facet functions <code>use_facet()</code> and <code>has_facet()</code> , and standard facet classes <code>num_get</code> , <code>collate</code> , <code>money_put</code> , <code>codecvt</code> , etc.
<locale>	<code>lconv</code> and the <code>setlocale()</code> and <code>localeconv()</code> functions. <code>setlocale()</code> only changes the C locale.
<regex>	Everything related to regular expressions.
<charconv> C++17	The <code>from_chars()</code> and <code>to_chars()</code> functions, and <code>chars_format</code> class.

Concurrency (Chapter 7)

Header	Contents
<thread>	The <code>thread</code> class and the <code>this_thread</code> namespace.
<future>	<code>future</code> and <code>shared_future</code> , <code>future_error</code> , and providers <code>promise</code> , <code>packaged_task</code> , and <code>async()</code> .
<mutex>	<code>mutex</code> , <code>recursive_mutex</code> , <code>timed_mutex</code> , <code>recursive_timed_mutex</code> , <code>lock_guard</code> , <code>unique_lock</code> , <code>scoped_lock</code> , and related types. Functions <code>try_lock()</code> , <code>lock()</code> , and <code>call_once()</code> .
<shared_mutex>	<code>shared_mutex</code> , <code>shared_timed_mutex</code> , and <code>shared_lock</code> .
<condition_variable>	<code>condition_variable</code> and <code>condition_variable_any</code> , and the function <code>notify_all_at_thread_exit()</code> .
<atomic>	Atomic types and fences.

Diagnostics (Chapter 8)

Header	Contents
<cassert>	The <code>assert()</code> macro.
<exception>	<code>exception</code> and <code>bad_exception</code> , exception pointers, nested exceptions, <code>terminate</code> and <code>unexpected</code> handlers.
<stdexcept>	Exception classes for reporting common errors: <code>logic_error</code> , <code>runtime_error</code> , and their generic subclasses.
<system_error>	The <code>std::system_error</code> exception used to report low-level errors, and the concepts of error codes, conditions, and categories.
<cerrno>	The <code>errno</code> expression, and default error condition values.

The C Standard Library

This section lists the remaining C headers that are not mentioned earlier.

Header	Contents
<ciso646>	<i>Only useful for C. Defines macros such as <code>and</code>, <code>or</code>, <code>not</code>, etc. In C++, those are reserved words, so this header is empty.</i>
<csetjmp>	<i><code>longjmp()</code> and <code>setjmp()</code>. Do not use these in C++.</i>
<csignal>	<i><code>signal()</code> and <code>raise()</code>. Do not use these in C++.</i>
<cstdlibalign>	<i>The <code>__alignas_is_defined</code> macro: always expands to 1 for C++. Deprecated since C++17.</i>
<cstdlibarg>	<i>The <code>va_list</code> type and functions <code>va_start()</code>, <code>va_arg()</code>, <code>va_end()</code>, and <code>va_copy()</code> to handle variable-length argument lists. In C++ it is recommended to use type-safe variadic templates instead.</i>
<cstdlibbool>	<i>The <code>__bool_true_false_are_defined</code> macro: expands to 1 for C++. Deprecated since C++17.</i>
<cstdlibdef>	<i>Types <code>ptrdiff_t</code>, <code>size_t</code>, <code>max_align_t</code>, and <code>nullptr_t</code>. The macro <code>offsetof()</code> and the constant <code>NULL</code>.</i>
<cstdliblib>	<i>String conversion functions: <code>atof()</code>, <code>strtof()</code>, etc. Multibyte character functions: <code>mblen()</code>, <code>mbtowc()</code>, and <code>wctomb()</code>. Multibyte string conversion: <code>mbstowcs()</code> and <code>wcstombs()</code>. Searching and sorting: <code>bsearch()</code> and <code>qsort()</code> (use <algorithm>). Random numbers: <code>rand()</code> and <code>srand()</code> (deprecated, use <random>). Memory management: <code>calloc()</code>, <code>free()</code>, <code>malloc()</code>, and <code>realloc()</code>. Integer functions: <code>abs()</code>, <code>div()</code>, <code>labs()</code>, <code>ldiv()</code>, <code>llabs()</code>, and <code>lldiv()</code>. Functions <code>abort()</code>, <code>atexit()</code>, <code>at_quick_exit()</code>, <code>exit()</code>, <code>getenv()</code>, <code>quick_exit()</code>, <code>system()</code>, and <code>_Exit()</code>.</i>

Index

`_1, _2, etc.`, 44
`"C" locale`, 200
`<<`, xxii, 157
`>>`, 157

■ A

`abs()`, 1
`absolute()`, 171
Absolute path, 163, 171
`accumulate()`, 132
`acos()`, 3
`acosh()`, 3
Active thread, 233
`add_const`, 67
`add_cv`, 67
`add_lvalue_reference`, 67
`add_pointer`, 67
`addressof`, 72
`add_rvalue_reference`, 67
`add_volatile`, 67
`adjacent_difference()`, 135
`adjacent_find()`, 118
`adjustfield`, 144
`adopt_lock`, 241, 242
`advance()`, 76
Aggregate, 132
`<algorithm>`, 115
Aliasing (`shared_ptr`), 41
`alignment_of`, 66
`allocate_shared`, 108
allocators, 108
`all_of()`, 117
`any`, 55
`any_cast()`, 56
`any_of()`, 117
`app`, 154
Appending, 79, 167
`apply()`, 35
Argument-dependent lookup (ADL), 32
Arithmetic type properties, 11
`array`, 84
ASCII, 189
`as_const`, 71
`asctime()`, 60
As-if rule, 249
`asin()`, 3
`asinh()`, 3
Assertions, 257
Associative containers
 ordered, 93
 unordered, 103
`assoc_laguerre()`, 7
`assoc_legendre()`, 7
`async()`, 235
Asynchronous programming, 235
 See also Futures
`atan()`, 3
`atan2()`, 3
`atanh()`, 3
`ate`, 154
`<atomic>`, 250
`atomic_flag`, 255
`atomic_signal_fence()`, 255
`atomic_thread_fence()`, 255
Atomic Variables, 250
 `compare_exchange()`, 252
 construction, 251
 `exchange()`, 252
 integral and pointer types, 251, 254
 `lock_free()`, 253
 non-member functions, 255

Atomic Variables (*cont.*)

- specializations, 251
- store() and load(), 252
- synchronization, 254
- auto_format, 164
- auto_ptr, 39
- available, 178
- Available disk space, 177

■ B

- back_inserter(), 139
- back_insert_iterator, 138
- bad_alloc, 233, 248, 258
- bad_any_cast, 56, 258
- bad_array_new_length, 258
- badbit, 147
- bad_cast, 258
- bad_exception, 258
- bad_function_call, 45, 258
- bad_optional_access, 48, 50, 258
- bad_typeid, 258
- bad_variant_access, 52, 258
- bad_weak_ptr, 258
- basefield, 144
- basic_string, 189
- basic_string_view, 195
- begin(), 75
- bernoulli_distribution, 19
- Bessel functions, 6
- beta(), 9
- Bidirectional iterator, 73
- bidirectional_iterator_tag, 74
- binary, 154
- binary_function, 42
- binary_search(), 119
- bind(), 44
- bind1st(), 42
- bind2nd(), 42
- Binding function arguments, 44
- binomial_distribution, 19
- bit_and, 43
- bit_not, 43
- bit_or, 43
- bitset, 89
- bit_xor, 43
- boolalpha, 144
- bool_constant, 63
- boyer_moore_horspool_searcher, 120
- boyer_moore_searcher, 120
- byte, 35

■ C

- call_once(), 245
- canonical(), 170
- Capacity, 80, 178
- CAS operations, 252
- <cassert>, 257
- cauchy_distribution, 19
- cbegin(), 75
- cbrt(), 2
- <cctype>, 195
- ceil(), 3
- cend(), 75
- cerr, 150
- <cerrno>, 262, 264
- C error numbers, 264
- Character classes, 196, 207
- Character classification, 196, 207
- Character-encoding conversion, 197
- Character encodings, 189, 197
- <charconv>, 226, 229
- chars_format, 229
- char16_t, 189
- char32_t, 189
- chi_squared_distribution, 19
- <chrono>, 56
- chrono_literals, 57
- cin, 152
- clamp(), 10
- classic(), 202
- <locale>, 213
- C locales, 213
- clock(), 60
- Clocks, 59
- CLOCKS_PER_SEC, 60
- clock_t, 60
- clog, 150
- cmatch, 219
- <cmath>, 1, 5
- codecvt, 198
- codecvt_byname, 199
- codecvt_utf8, 198
- codecvt_utf16, 198
- codecvt_utf8_utf16, 198
- collate, 208
- common_type, 67
- Compare-and-swap, 252
- comp_ellint_1(), 7
- comp_ellint_2(), 8
- <complex>, 13
- complex_literals, 13

Complex numbers, 13
Concatenation, 79, 167
conditional, 67
condition_variable, 246
condition_variable_any, 246
Condition Variables, 246
 exceptions, 248
 notification, 247
 synchronization, 249
 timeouts, 247
 waiting, 246
conjunction, 70
constexpr if, 68
Container adaptors, 91
Containers, 73
copy(), 123, 176
copy_backward(), 123
copyfmt_event, 158
copy_if(), 123
copy_n(), 123
copy_options, 177
copysign(), 4
copy_symlink(), 169
copy_symlinks, 177
cos(), 3
cosh(), 3
count(), 117
count_if(), 117
cout, 150
crbegin(), 75
create_directories(), 176
create_directory(), 176
create_directory_symlink(), 169
create_hard_link(), 169
create_hard_links, 177
create_symlink(), 169
create_symlinks, 177
cref(), 43, 46
crend(), 75
Critical section, 249 *See also* Mutexes
<cstdlib>, 10
<cstdio>, 180, 181
C-style date and time utilities, 60
csub_match, 219
ctime(), 60
ctype, 207
Cumulative sum, 134
Currency symbol, 205
current_exception(), 259
current_path(), 170
Current thread, 233

<cwctype>, 195
cyl_bessel_i(), 6
cyl_bessel_j(), 6
cyl_bessel_k(), 6
cyl_neuman(), 6

■ **D**

data() (non-member function), 89
Data race, 137, 238, 250
Date formatting, 206
Date utilities, 60
Deadlock, 137, 233, 240, 244
dec, 144
decay, 67
Decimal separator, 203
declval(), 70
defaultfloat, 145
default_random_engine, 17
default_searcher, 120
defer_lock, 242
deque, 83
destroy(), 136
destroy_at(), 136
destroy_n(), 136
Difference, 130
difftime(), 60
Digit grouping, 203
Directories, 162, 178
directories_only, 177
directory_entry, 178
directory_iterator, 178
Directory listing, 178
directory_options, 179
Directory separator, 162
discard_block_engine, 16
discrete_distribution, 20
disjunction, 70
Disk space, 177
distance(), 76
Distribution, *see* Random number distributions
div(), 2
divides, 43
domain_error, 258
Dot product, 133, 160
Double-checked locking, 245
Double-ended queue, 83
Doubly linked list, 84
duration, 57
duration_cast(), 57

■ E

ECMAScript grammar, *see* Regular expressions

ellint_1(), 7
 ellint_2(), 7
 Emplacement, 79, 95–96
 empty() (non-member function), 89
 enable_if, 68
 enable_shared_from_this, 41
 end(), 75
 endl, 150
 ends, 150
 eofbit, 147
 Epoch, 58
 Epsilon, 12
 equal(), 122
 equal_range(), 119
 equal_to, 43
 equivalent(), 172
 erase_event, 158
 erf(), erfc(), 8
 errc, 229, 262
 errno, 264
 error_category, 262
 error_code, 163, 229, 262
 error_condition, 262
 Error functions, 8
 Exception pointers, 259
 exception_ptr, 259
 Exceptions, 148, 163
 Exceptions class hierarchy, 258
 exchange(), 32
 exclusive_scan(), 134
 execution namespace, 136
 exists(), 173
 exp(), 2
 exp2(), 2
 expint(), 8
 expm1(), 2
 exponential_distribution, 20
 Extension, *see* File extension
 extent, 67
 extreme_value_distribution, 20

■ F

fabs(), fabsf(), fabsl(), 1
 Facets, 202 *See also* Localization
 failbit, 147
 failure, 258

False sharing, 248
 false_type, 63, 69
 fdim(), 2
 Fences, 249, 254, 255
 File extension, 166, 168
 File links, 168
 File permissions, 174
 Files, 162
 file_size(), 177
 file_status, 172
 File streams, 155
 <filesystem>, 162
 filesystem_error, 163
 File types, 173
 fill(), 122
 Fill character, 145, 149
 fill_n(), 122
 find(), 118
 find_end(), 120
 find_first_of(), 118
 find_if(), 117
 find_if_not(), 117
 Fire-and-forget, 232, 236
 First-in first-out (FIFO), 91
 fisher_f_distribution, 19
 fixed, 144
 Fixed-width integer types, 10
 floatfield, 144
 Floating-point numbers
 Epsilon, 12
 Infinity, 12
 NaN, 2, 4, 13
 floor(), 3
 flush, 150
 fma(), 2
 fmax(), 2
 fmin(), 2
 fmod(), 1
 fmtflags, 144
 Fold, 132
 follow_directory_symlink, 179
 for_each(), 115
 for_each_n(), 116
 Formatting, *See also* to_string(),
 to_chars(), printf(),
 Regular expressions,
 and Stream I/O
 date, 61, 206
 monetary, 146
 numerical, 144
 time, 61, 206

forward(), 31
 forward_as_tuple(), 35
 Forwarding reference, 31
 Forward iterator, 73
 forward_iterator_tag, 75
 forward_list, 84
 fpclassify(), 4
 FP_INFINITE, 4
 FP_NAN, 4
 FP_NORMAL, 4
 fpos, 142
 fprintf(), 181
 FP_SUBNORMAL, 4
 FP_ZERO, 4
 free, 178
 Free space, 177
 frexp(), 3
 from_chars(), 229
 front_inserter(), 139
 front_insert_iterator, 138
 fscanf(), 185
 <fstream>, 155
 function, 45
 <functional>, 42, 71
 Function object, 42
 for class members, 46
 Functor, 42
 <future>, 234
 future_errc, 238
 future_error, 237, 258
 Futures, 234
 exceptions, 237
 providers, 234, 235
 async(), 235
 packaged tasks, 236
 promises, 237
 shared state, 234
 synchronization, 250

■ G

gamma_distribution, 20
 Gamma functions, 8
 gcd(), 4
 generate(), 122
 generate_canonical, 19
 generate_n(), 122
 Generic function wrappers, 45
 generic_format, 162
 geometric_distribution, 19
 get_default_resource(), 109

get_if() (variant), 53
 getline(), 151, 153
 get_money(), 146
 get_terminate(), 266
 get_time(), 146
 get() (tuple), 34
 get() (variant), 52
 global(), 202
 gmtime(), 60
 goodbit, 147
 Grammar
 printf(), 181
 regular expressions, 214
 scanf(), 185
 time and date formatting, 61
 greater, 43
 greater_equal, 43
 gsllice, 25

■ H

hard_link_count(), 169
 hardware_concurrency(), 233
 hardware_constructive_
 interference_size, 248
 hardware_destructive_
 interference_size, 248
 has_facet(), 202
 hash, 105
 Hash functions, 104
 Hash map, 104
 has_unique_object_representations, 66
 Header
 <algorithm>, 115
 <any>, 55
 <array>, 84
 <atomic>, 250
 <bitset>, 89
 <cassert>, 257
 <cctype>, 195
 <cerrno>, 262, 264
 <charconv>, 226, 229
 <chrono>, 56
 <locale>, 213
 <cmath>, 1, 5
 <codecvt>, 197
 <complex>, 13
 <condition_variable>, 246
 <cstdlib>, 10
 <cstdio>, 180, 181
 <ctime>, 60

Header (*cont.*)

- <cwctype>, 195
- <deque>, 83
- <exception>, 258
- <filesystem>, 162
- <forward_list>, 84
- <fstream>, 155
- <functional>, 42, 71
- <future>, 234
- <initializer_list>, 47
- <iomanip>, 145
- <ios>, 142, 143
- <iosfwd>, 142
- <iostream>, 142, 150
- <istream>, 151
- <iterator>, 73, 113, 138
- <limits>, 11
- <list>, 84
- <locale>, 200
- <map>, 94, 98
- <memory>, 36, 108
- <memory_resource>, 108
- <mutex>, 238
- <numeric>, 132
- <optional>, 48
- <ostream>, 149
- <queue>, 91
- <random>, 15
- <ratio>, 14
- <regex>, 214
- <scoped_allocator>, 111
- <set>, 98
- <shared_mutex>, 240
- <sstream>, 153
- <stack>, 92
- <stdexcept>, 258
- <streambuf>, 161
- <string>, 189, 227
- <string_view>, 194
- <system_error>, 262
- <thread>, 231
- <tuple>, 34
- <typeid>, 62
- <typeinfo>, 62
- <type_traits>, 63
- <unordered_map>, 103
- <unordered_set>, 103
- <utility>, 29, 33
- <valarray>, 23
- variant, 50
- <vector>, 76

- Heaps, 131
- hermite(), 7
- hex, 144
- hexfloat, 145
- high_resolution_clock, 59
- Hints, 97
- holds_alternative(), 52
- hypot(), 2

■ I, J

- ifstream, 155
- ilogb(), 3
- imbue_event, 158
- in, 154
- includes(), 129
- inclusive_scan(), 134
- independent_bits_engine, 17
- indirect_array, 27
- Infinity, 12
- <initializer_list>, 47
- Initializer-list constructors, 47
- inner_product(), 133
- in_place_index, in_place_index_t, 51
- inplace_merge(), 129
- in_place_type, in_place_type_t, 51-52, 55
- Input iterators, 113
- input_iterator_tag, 113
- Input streams, *see* Stream I/O
- inserter(), 139
- insert_iterator, 139
- integral_constant, 63
- Integrals, 7, 8
- int_fastX_t, 10
- int_leastX_t, 10
- internal, 144
- Internationalization, *see* Localization
- Intersection, 130
- intmax_t, 11
- intptr_t, 11
- intX_t, 10
- invalid_argument, 258
- invoke, 71
- invoke_result, 69
- I/O, *see* Stream I/O
- <iomanip>, 145
- I/O Manipulator, *see* Stream I/O
- I18n, *see* Localization
- <ios>, 142, 143
- ios_base, 143
- <iosfwd>, 142

iostate, 147
 ostream, 142, 150, 153
 iota(), 123
 is_abstract, 65
 is_aggregate, 66
 is_alnum(), 196, 207
 is_alpha(), 196, 207
 is_arithmetic, 64
 is_array, 64
 is_assignable, 65
 is_base_of, 66
 is_blank(), 196, 215
 is_block_file(), 173
 is_character_file(), 175
 is_class, 64
 is_cntrl(), 196, 215
 is_const, 65
 is_constructible, 65
 is_convertible, 66
 is_copy_assignable, 65
 is_copy_constructible, 65
 is_default_constructible, 65
 is_destructible, 65
 is_digit(), 196, 207
 is_directory(), 173
 is_empty, 66
 is_enum, 64
 is_error_code_enum, 263
 is_error_condition_enum, 264
 is_execution_policy_v, 137
 is_fifo(), 173
 is_final, 65
 is_finite(), 4
 is_floating_point, 64
 is_function, 64
 is_fundamental, 64
 is_graph(), 196, 207
 isgreater(), 4
 isgreaterequal(), 4
 is_heap(), 131
 is_heap_until(), 132
 isinf(), 4
 is_integral, 64
 is_invocable, 69
 is_invocable_r, 69
 isless(), 4
 islessequal(), 4
 islessgreater(), 4
 is_literal_type, 66
 is_lower(), 196, 208
 is_lvalue_reference, 64
 is_member_function_pointer, 64
 is_member_object_pointer, 64
 is_member_pointer, 64
 is_move_assignable, 65
 is_move_constructible, 65
 isnan(), 4
 isnormal(), 4
 is_null_pointer, 64
 is_object, 64
 is_other(), 173
 is_partitioned(), 126
 is_permutation(), 130
 is_pod, 66
 is_pointer, 64
 is_polymorphic, 65
 is_print(), 196, 208
 is_punct(), 196, 207
 is_reference, 64
 is_regular_file(), 173
 is_rvalue_reference, 64
 is_same, 66
 is_scalar, 64
 is_signed, 65
 is_socket(), 173
 is_sorted(), 127
 is_sorted_until(), 127
 is_space(), 196, 208
 is_standard_layout, 66
 is_swappable, is_swappable_with, 65
 is_symlink(), 169, 173
 <istream>, 151
 istream_iterator, 160
 istringstream, 153
 is_trivial, 66
 is_trivially_copyable, 65, 68
 is_union, 64
 isunordered(), 4
 is_unsigned, 65
 is_upper(), 196, 208
 is_void, 64
 is_volatile, 65
 is_walnum(), 196
 is_walpha(), 196
 is_wblank(), 196
 is_wcntrl(), 196
 is_wdigit(), 196
 is_wgraph(), 196
 is_wlower(), 196
 is_wprint(), 196
 is_wpunct(), 196
 is_wspace(), 196

■ INDEX

`is_wupper()`, 196
`is_wxdigit()`, 196
`is_xdigit()`, 196, 215
`iterator`, 73, 113
Iterator adaptors, 138
Iterators
 bidirectional, 73
 categories, 73, 113
 forward, 73
 input, 113
 output, 113
 random access, 73
 reverse, 74, 75
 stream iterators, 160
Iterator tags, 74
`iterator_traits`, 75
`iter_swap()`, 124

■ K

`knuth_b`, 17

■ L

`labs()`, 1
`laguerre()`, 7
Last-in first-out (LIFO), 92
Launch policy, 236
Lazy initialization, 245
`LC_ALL`, 213
`LC_COLLATE`, 213
`lcm()`, 4
`LC_MONETARY`, 213
`LC_NUMERIC`, 213
`lconv`, 213
`LC_TIME`, 213
`LC_CTYPE`, 213
L1 data cache line size, 248
`ldexp()`, 4
`ldiv()`, 2
`left`, 144
Left fold, 132
`legendre()`, 7
`length_error`, 258
`less`, 43
`less_equal`, 43
`lexicographical_compare()`, 127
`lgamma()`, 8
`<limits>`, 11
`linear_congruential_engine`, 16

Line-by-line input, 153
`list`, 84
List-specific algorithms, 85
`llabs()`, 1
`lldiv()`, 2
`llrint()`, 3
`llround()`, 3
`localeconv()`, 213
Locale names, 200
Localization, 200
 C locales, 213
 combining facets, 210
 custom facets, 211
 global locale, 201
 locale facet categories, 203
 locale facets, 202
 locale names, 200
 standard facets, 203
 character classification and
 transformation, 207
 character-encoding
 conversions, 208
 formatting and parsing
 monetary values, 205
 numeric values, 204
 time and dates, 206
 message retrieval, 209
 monetary punctuation, 204
 numeric punctuation, 203
 string ordering and
 hashing, 208
 std::locale, 200
`localtime()`, 60
`lock()`, 244
`lock_guard`, 241, 276
Lock-free data structures, 252
Locks, *see* Mutexes
`log()`, 2
`log2()`, 2
`log10()`, 2
`logb()`, 3
`log1p()`, 2
`logical_and`, 43
`logical_not`, 43
`logical_or`, 43
`logic_error`, 258
`lognormal_distribution`, 19
`lower_bound()`, 119
`lrint()`, 3
`lround()`, 3

■ **M**

- Magic statics, 245
- main(), xxii
- make_error_code(), 263
- make_error_condition(), 264
- make_exception_ptr(), 259
- make_from_tuple, 35
- make_heap(), 131
- make_move_iterator(), 138
- make_optional(), 49
- make_pair(), 33
- make_ready_at_thread_exit(), 237
- make_reverse_iterator(), 138
- make_shared(), 40
- make_signed, 67
- make_tuple(), 34
- make_unique(), 37–38
- make_unsigned, 67
- Manipulator, *see* Stream I/O
- map, 94
- mask_array, 26
- match_flag_type, 218, 224
- match_results, 219, 220
- Mathematical functions
 - classification, 4
 - common functions, 1
 - comparison, 4
 - error functions, 8
 - error handling, 5
 - exponential functions, 2
 - floating-point manipulation, 3
 - gamma functions, 8
 - hyperbolic functions, 3
 - logarithmic functions, 2
 - power functions, 2
 - rounding, 3
 - special functions, 5
 - trigonometric functions, 3
- MATH_ERREXCEPT, 5
- math_errhandling, 5
- MATH_ERRNO, 4
- max(), 9
- max_element(), 118
- Maximum representable
 - number, 11
- mbstate_t, 197
- Member function object, 46
- memcpy(), 68
- mem_fn(), 46
- mem_fun(), 42
- mem_fun_ref(), 42
- <memory>, 36, 108
- Memory model, 249
- Memory pool, 110
- memory_order, 254
- memory_resource, 108
- merge(), 129
- Merging, *See* Appending,
 - Concatenation
- Merging (associative containers), 100
- mersenne_twister_engine, 16
- messages, 209
- min(), 9
- min_element(), 118
- Minimum representable number, 11
- minmax(), 10
- minmax_element(), 118
- minstd_rand, 17
- minstd_rand0, 17
- minus, 43
- mismatch(), 122
- mktime(), 60
- modf(), 3
- modulus, 43
- Monetary formatting, 146
- money_get, 205
- money_punct, 204, 211
- money_punct_byname, 212
- money_put, 205
- monostate, 51
- monotonic_buffer_resource, 110
- move(), 29, 123
- move_backward(), 123
- move_if_noexcept(), 30
- move_iterator, 138
- Move semantics, 29
- mt19937, 17
- mt19937_64, 17
- multimap, 98
- multiplies, 43
- multiset, 98
- mutex, 238, 239
- Mutexes, 238
 - critical section, 249
 - exceptions, 244
 - lock types
 - lock_guard, 241
 - scoped_lock, 239, 241
 - shared_lock, 243
 - unique_lock, 242
 - locking, 238, 241

Mutexes (*cont.*)

- locking multiple mutexes, 244
- native_handle(), 240
- RAII, 238, 241
- readers-writers, 240
- recursion, 240
- reentry, 240
- sharing ownership, 240
- synchronization, 249
- timeouts, 240

■ N

- NaN, 2, 4, 13
- nan(), nanf(), nanl(), 2
- native_format, 164
- NDEBUG, 257
- nearbyint(), 3
- negate, 43
- negation, 70
- negative_binomial_distribution, 19
- nested_exception, 260
- Neutral locale, 200
- new_delete_resource(), 109
- next(), 76
- nextafter(), 4
- next_permutation(), 131
- nexttoward(), 4
- Nodes (associative containers), 100
- none_of(), 117
- normal_distribution, 19
- not1(), 45
- not2(), 45
- not_equal_to, 43
- not_fn(), 45
- notify_all_at_thread_exit(), 247
- npos, 190
- nth_element(), 18
- null_memory_resource(), 109
- nullopt, nullopt_t, 48
- <numeric>, 132
- Numeric conversions, 226
- numeric_limits, 11
- Numerical formatting, 145
- num_get, 204
- numpunct, 203, 212
- numpunct_byname, 212
- num_put, 204

■ O

- oct, 144
- ofstream, 155
- once_flag, 245
- openmode, 154, 155
- optional, 48
- Ordered associative containers, 93
- <ostream>, 149
- ostream_iterator, 160
- ostringstream, 154
- out, 154
- out_of_range, 258
- Output iterators, 113
- output_iterator_tag, 113
- Output streams, *see* Stream I/O
- overflow_error, 258
- overwrite_existing, 177
- owner_less, 41

■ P

- packaged_task, 236
- pair, 33
- par, 137
- Parallel algorithms, 136
- parallel_policy, 137
- parallel_unsequenced_policy, 137
- Parsing, *see* stoi(), stof(), from_chars(), scanf(), Regular expressions, and Stream I/O
- Parsing floating-point numbers, 228
- Parsing integers, 227
- partial_sort(), 127
- partial_sort_copy(), 127
- partial_sum(), 134
- partition(), 126
- partition_copy(), 126
- partition_point(), 127
- par_unseq, 137, 138
- path, 164
- Pathnames, 162, 166
- Perfect forwarding, 31
- Permissions, 174
- permissions(), 175
- perms, 174
- Permutations, 130
- pererror(), 265
- Person class, xxiii
- Piecewise construction, 34, 96

piecewise_constant_distribution, 21
 piecewise_linear_distribution, 22
 Placeholders, 44
 plus, 43
 pmr (namespace), 108, 109
 poisson_distribution, 20
 polymorphic_allocator, 108
 Polynomials, 7
 pool_options, 111
 pop_heap(), 131
 POSIX error codes, 264, 267
 pow(), 2
 Predefined functors, 43
 preferred_separator, 162
 Prefix sum, 134
 prev(), 76
 prev_permutation(), 13
 printf(), 181

- conversion specifiers, 182
- flags, 184
- formatting syntax, 183
- length modifiers, 185

 priority_queue, 91
 promise, 237
 proximate(), 171
 ptr_fun(), 42
 push_heap(), 131
 put_money(), 146
 put_time(), 146

■ Q

queue, 91
 quoted(), 146

■ R

RAII, 36, 241
 rand(), 15
 <random>, 15
 Random access iterator, 74
 Random number distributions, 18

- Bernoulli, 19
- Normal, 19
- Poisson, 20
- Sampling
 - Discrete, 20
 - Piecewise constant, 21
 - Piecewise linear, 22
- Uniform, 19

Random number generators, 15

- Non-deterministic, 18
- Pseudorandom number engines, 16
 - Engine adaptors, 16
 - Predefined engines, 17

 Random numbers, 15

- Seeding, 18

 random_access_iterator_tag, 75
 random_device, 18
 range_error, 258
 rank, 66
 ranlux24, 17
 ranlux24_base, 17
 ranlux48, 17
 ranlux48_base, 17
 <ratio>, 14
 ratio_add, 14
 ratio_divide, 14
 ratio_equal, 14
 ratio_multiply, 14
 Rational numbers, 14
 ratio_subtract, 14
 rbegin(), 75
 Readers-writers locks, *see* Mutexes
 read_symlink(), 169
 recursive, 177
 recursive_directory_entry, 179
 recursive_mutex, 239
 recursive_timed_mutex, 239
 reduce(), 132
 ref(), 43, 46
 Reference wrappers, 43, 46
 reference_wrapper, 43, 46
 regex, 214, 216
 regex_error, 216, 219, 223
 regex_iterator, 221
 regex_match(), 218
 regex_replace(), 223
 regex_search(), 218
 regex_token_iterator, 222
 Regular expressions, 214

- grammar, 214
 - assertions, 214, 225
 - atoms, 214, 224
 - back reference, 214
 - character classes, 215
 - disjunction, 214
 - greediness, 216
 - lookahead, 226
 - quantifiers, 226

Regular expressions (*cont.*)
 grammar options, 216
 matching and searching patterns, 218
 match iterators, 221
 match results, 219
 raw string literals, 214
 replacing patterns, 223
`std::regex`, 216

Relational operators, 36

`relative()`, 171

Relative path, 163, 171

`release()` (`unique_ptr`), 38

`rel_ops`, 36

`remainder()`, 1

`remove()`, 176

`remove()` (algorithm), 124, 275

`remove()` (file), 176, 180

`remove_all()`, 176

`remove_const`, 67

`remove_copy()`, 125

`remove_copy_if()`, 125

`remove_cv`, 67

Remove-erase idiom, 81, 124

`remove_if()`, 124

`remove_lvalue_reference`, 67

`remove_pointer`, 67

`remove_rvalue_reference`, 67

`remove_volatile`, 67

`remquo()`, 1

`rename()`, 176, 180

`rend()`, 75

`replace()`, 125

`replace_copy()`, 125

`replace_copy_if()`, 125

`replace_if()`, 125

`reset()` (`shared_ptr`), 40

`reset()` (`unique_ptr`), 38

`resetiosflags()`, 145

`resize_file()`, 177

Resource Acquisition Is
 Initialization, *see* RAII

`result_of`, 70

`rethrow_exception()`, 259

`rethrow_if_nested()`, 261

`reverse()`, 126

`reverse_copy()`, 126

Reverse iterator, 74, 75

`reverse_iterator`, 138

`riemann_zeta()`, 9

Right fold, 132

`right`, 144

`rint()`, 3

`rotate()`, 126

`rotate_copy()`, 126

`round()`, 3

`runtime_error`, 258

Runtime type identification, 62

■ S

`sample()`, 128

Scalar product, 133

`scalbln()`, `scalbn()`, 4

Scan, 134

`scanf()`, 185
 conversion specifiers, 186
 formatting syntax, 186
 length modifiers, 187

scientific, 144

`<scoped_allocator>`, 111

`scoped_allocator_adaptor`, 111

`scoped_lock`, 239, 241

`search()`, 120

`search_n()`, 120

`seekdir`, 149

Selection algorithm, 128

`seq`, 137

Sequence comparison, 121

`sequenced_policy`, 137

Sequential containers, 76

`set`, 98

`setbase()`, 145

`set_default_resource()`, 109

`set_difference()`, 130

`set_exception_at_thread_exit()`, 237

`setfill()`, 145

`set_intersection()`, 130

`setiosflags()`, 145

`setlocale()`, 213

`setprecision()`, 145

`set_symmetric_difference()`, 130

`set_terminate()`, 266

`set_union()`, 130

`set_value_at_thread_exit()`, 237

`setw()`, 146

SFINAE, 68

`shared_from_this()`, 41

`shared_future`, 234

`shared_lock`, 243

`shared_mutex`, 239, 276

`shared_ptr`, 39

`shared_timed_mutex`, 239

- showbase, 144
- showpoint, 144
- showpos, 144
- shuffle(), 128
- shuffle_order_engine, 17
- signbit(), 4
- sin(), 3
- Singleton, 245
- sinh(), 3
- SI ratios, 4
- size() (non-member function), 89
- skip_existing, 177
- skip_permission_denied, 179
- skip_symlinks, 177
- skipws, 144
- sleep_for(), 233
- Sleeping, 233
- sleep_until(), 233
- slice, 24
- Smart pointers, 36 *See also* RAII.
- smatch, 219
- sort(), 128
- sort_heap(), 131
- space(), 178
- space_info, 178
- sph_bessel(), 6
- sph_legendre(), 7
- sph_neuman(), 6
- Splicing, 85
- Splitting strings, *see*
 - regex_token_iterator
- sprintf(), 181
- Spurious wakeups, 246
- sqrt(), 2
- sscanf(), 185
- <sstream>, 153
- ssub_match, 219
- stable_partition(), 126
- stable_sort(), 127
- stack, 92
- Standard Template Library, xix
- status(), 172
- status_known(), 173
- std, xxi
- stderr, 150
- <stdexcept>, 258
- stdin, 152
- stdout, 150
- steady_clock, 59
- STL, xix
- stod(), 228
- stof(), 228
- stoi(), 227
- stol(), 227
- stold(), 228
- stoll(), 227
- stoul(), 227
- stoull(), 227
- <streambuf>, 161
- Stream Buffers, 161
- Stream I/O
 - class hierarchy, 141
 - default initialization, 145, 148
 - error handling, 148
 - file streams, 155
 - formatting flags, 144
 - helper types, 142
 - input streams, 151
 - I/O manipulators, 145, 150
 - open modes, 155
 - output streams, 149
 - standard input streams, 152
 - standard output streams, 150
 - redirect, 161
 - state bits, 148
 - stream iterators, 160
 - string streams, 153
 - thread safety, 151
- streamoff, 142
- streamsize, 142
- strerror(), 265
- strftime(), 60
- <string>, 189, 227
- string literals, 192
- Strings, 189
 - comparing, 193
 - constructing, 192
 - formatting (*see* Formatting)
 - length, 192
 - modifying, 191
 - npos, 190
 - parsing (*see* Parsing)
 - searching, 190
 - string literal operator, 192
 - substrings, 193
 - types, 189
- String streams, 153
- string_view, 194
- string_view literals, 195
- student_t_distribution, 19
- sub_match, 219
- Subsequence search, 120

■ INDEX

Substrings, 193
subtract_with_carry_engine, 16
Summation, 132
sv ("sv"), 195
swap(), 32
swap_ranges(), 123
Symbolic link, 168
symlink_status(), 172
Symmetric difference, 130
Synchronization, 249 *See also* Memory model
synchronized_pool_resource, 110
sync_with_stdio(), 151, 152
system_clock, 59
system_error, 163, 229, 233, 237, 238, 244, 248, 258

■ T

tan(), 3
tanh(), 3
terminate(), 266
tgamma(), 8
Thousands separator, 203
<thread>, 231
Threads, 231
 exceptions, 233
 fire-and-forget, 232, 236
 identifiers, 232
 joining, 232
 launching, 231
 sleeping, 233
 synchronizing, 250
 yielding, 233
throw_with_nested(), 261
tie(), 34
time(), 60
timed_mutex, 239
Time formatting, 146
time_get, 206
time_point, 58
time_point_cast(), 58
time_put, 206
time_t, 60
Time utilities, 56
tm, 60
tmpfile(), 180
tmpnam(), 180
to_chars(), 229
Tokenizing, *see* regex_token_iterator
tolower(), 196, 208

to_string(), 90
toupper(), 196, 208
Torn reads and writes, 252 *See also* Atomic variables
tolower(), 196
to_wstring(), 228
toupper(), 196
transform(), 116
transform_exclusive_scan(), 135
transform_inclusive_scan(), 135
transform_reduce(), 132, 133
Transparent operator functors, 44, 99
True sharing, 248
true_type, 63, 69
trunc, 154
trunc(), 3
try_lock(), 244
try_to_lock, 242
<tuple>, 34
tuple_element, 35
tuple_size, 35
Type classification, 64
typeid(), 62
<typeid>, 62
<typeinfo>, 62
Type properties, 65
Type property queries, 66
Type relationships, 66
Type traits, 63
<type_traits>, 63
Type transformations, 67

■ U

u16string, 189
u16string_view, 195
u32string, 189
u32string_view, 195
u8path(), 164
(u)int_fastX_t, 10
(u)int_leastX_t, 10
(u)intmax_t, 11
uintptr_t, 11
(u)intX_t, 10
unary_function, 42
uncaught_exceptions(), 265, 266
underflow_error, 258
underlying_type, 67
Unicode, 192, 197
uniform_int_distribution, 19
uniform_real_distribution, 19

uninitialized_copy(), 136
 uninitialized_copy_n(), 136
 uninitialized_default_construct(), 135
 uninitialized_fill(), 136
 uninitialized_fill_n(), 136
 uninitialized_move(), 136
 uninitialized_move_n(), 136
 uninitialized_value_construct(), 135
 Union, 130
 unique(), 125
 unique_copy(), 125
 unique_lock, 242
 unique_ptr, 36
 unitbuf, 144
 Universal reference, 31
 Unordered associative containers, 103
 unordered_map, 103
 unordered_multimap, 103
 unordered_multiset, 103
 unordered_set, 103
 unsynchronized_pool_resource, 110
 update_existing, 177
 upper_bound(), 119
 uppercase, 144
 Upstream memory resource, 110
 use_facet(), 202
 UTF-8, UTF-16, UTF-32, 189, 197
 <utility>, 29, 33, 272

■ **V**

<valarray>, 23
 variant, 50
 variant_alternative,
 variant_alternative_t, 54
 variant_size, variant_size_v, 54

vector, 76
 vector<bool>, 82
 visit(), 53
 void_t, 69

■ **W, X**

wbuffer_convert, 199
 wcerr, 150
 wchar_t, 189
 wcin, 152
 wclog, 150
 wcmatch, 219
 wcout, 150
 wcs_sub_match, 219
 weak_from_this(), 41
 weakly_canonical(), 170
 weak_ptr, 41
 Weak reference, 41
 weibull_distribution, 20
 Working directory, 170
 ws, 152
 wsmatch, 219
 wssub_match, 219
 wstring, 189
 wstring_convert, 198
 wstring_view, 195

■ **Y**

yield(), 233

■ **Z**

Zeta functions, 9