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

(*continued*)

P. Van Weert and M. Gregoire, *C++17 Standard Library Quick Reference*,
https://doi.org/10.1007/978-1-4842-4923-9

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

General Utilities (Chapter 2)

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

(*continued*)

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

Containers (Chapter 3)

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

(*continued*)

Header	Contents
<memory_resource> C++17	The polymorphic allocators and memory resources.
<scoped_allocator> C++17	The scoped_allocator_adaptor class template.

Algorithms (Chapter 4)

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

Input/Output (Chapter 5)

Header	Contents
<ios>	ios_base, basic_ios, and fpos, type aliases ios and wios, and types streamoff, streampos, wstreampos, and streamsize. Nonparametric I/O manipulators such as boolalpha, dec, scientific, etc.
<iomanip>	Parametric I/O manipulators, such as setbase(), setfill(), get_money(), put_time(), and more.
<ostream>	basic_ostream, and type aliases ostream and wostream. The endl, ends, and flush output manipulators.
<istream>	basic_istream and basic_iostream, and type aliases istream, wistream, iostream, and wiostream. The ws 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 <cstdint>. 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.*
<cwchar>	*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>: iswdigit(), iswspace(), towlower(), towupper(), etc.
<codecvt>	Unicode character-encoding conversion facets: codecvt_utf8, codecvt_utf16, and codecvt_utf8_utf16. Deprecated since C++17.
<cuchar>	*Functions to convert between 16- or 32-bit character and multibyte sequences: c16rtomb(), c32rtomb(), mbrtoc16(), mbrtoc32().*
<locale>	The locale class, overloads of <cctype> functions accepting a given locale, facet functions use_facet() and has_facet(), and standard facet classes num_get, collate, money_put, codecvt, etc.
<clocale>	lconv and the setlocale() and localeconv() functions. setlocale() only changes the C locale.
<regex>	Everything related to regular expressions.
<charconv> C++17	The from_chars() and to_chars() functions, and chars_format class.

Concurrency (Chapter 7)

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

Diagnostics (Chapter 8)

Header	Contents
<cassert>	The assert() macro.
<exception>	exception and bad_exception, exception pointers, nested exceptions, terminate and unexpected handlers.
<stdexcept>	Exception classes for reporting common errors: logic_error, runtime_error, and their generic subclasses.
<system_error>	The std::system_error exception used to report low-level errors, and the concepts of error codes, conditions, and categories.
<cerrno>	The errno 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>	*Only useful for C. Defines macros such as and, or, not, etc. In C++, those are reserved words, so this header is empty.*
<csetjmp>	*longjmp() and setjmp(). Do not use these in C++.*
<csignal>	*signal() and raise(). Do not use these in C++.*
<cstdalign>	*The __alignas_is_defined macro: always expands to 1 for C++. Deprecated since C++17.*
<cstdarg>	*The va_list type and functions va_start(), va_arg(), va_end(), and va_copy() to handle variable-length argument lists. In C++ it is recommended to use type-safe variadic templates instead.*
<cstdbool>	*The __bool_true_false_are_defined macro: expands to 1 for C++. Deprecated since C++17.*
<cstddef>	*Types ptrdiff_t, size_t, max_align_t, and nullptr_t. The macro offsetof() and the constant NULL.*
<cstdlib>	*String conversion functions: atof(), strtof(), etc.* *Multibyte character functions: mblen(), mbtowc(), and wctomb().* *Multibyte string conversion: mbstowcs() and wcstombs().* *Searching and sorting: bsearch() and qsort() (use <algorithm>).* *Random numbers: rand() and srand() (deprecated, use <random>).* *Memory management: calloc(), free(), malloc(), and realloc().* *Integer functions: abs(), div(), labs(), ldiv(), llabs(), and lldiv().* *Functions abort(), atexit(), at_quick_exit(), exit(), getenv(), quick_exit(), system(), and _Exit().*

Index

P. Van Weert and M. Gregoire, *C++17 Standard Library Quick Reference*,
https://doi.org/10.1007/978-1-4842-4923-9

B

C

D

E

F

G

H

I, J

K

L

M

N

O

P

Q

R

S

T

U

V

W, X

Y

Z