75 STL HEADERS IN UNDER 10 MINUTES

Excluding...

- C++23
- Deprecated Headers
- Headers from the C standard library

`<algorithm>`

- Computer science algorithms, mostly operating on a sequence of elements
- Overloaded on `[It, It)`
- Overloaded on `range` and `[It, S)` (C++20)

<any>

(C++17)

- `std::any`
- Type-erasure wrapper for any copy constructible type

`<array>`

(C++11)

- `std::array<T, N>`
- Container that encapsulates a fixed size C-style array
- Provides stronger type-safety

`<atomic>`

(C++11)

- `std::atomic<T>`
- Well defined concurrent access from multiple threads

`<barrier>`

(C++20)

- `std::barrier<CompletionFunction>`
- Reusable thread barrier that blocks up to N threads
- Executes `CompletionFunction` and opens the barrier when the last thread arrived

`<bit>`

(C++20)

- `enum class endian`
- Functions to access, manipulate and process individual bits and bit sequences (e.g. `has_single_bit, bit_ceil`)

`<bitset>`

- `std::bitset<N>`
- Fixed-size sequence of N bits
- Functions to access and manipulate the bits

`<charconv>`

(C++17)

- `from_chars, to_chars`
- Conversion between a `char`-sequence and a floating point or integer

<chrono>

(C++11)

- Date and time library, mainly consisting of clocks, time points and durations
- Calendar types and time zones (C++20)

`<compare>`

(C++20)

- Library support for the 3-way comparison `<=>`

`<complex>`

- `std::complex<T>`
- Complex number: `3.5 + 7i`

`<concepts>`

(C++20)

- Defines common concepts
- `same_as, integral`
- `movable, regular`
- `invocable, predicate`

`<condition_variable>`

(C++11)

- `std::condition_variable`, `std::condition_variable_any`
- Synchronization primitive to block threads until getting notified

`<coroutine>`

(C++20)

- Library support for coroutines
- `coroutine_handle<T>`, non-owning handle to an actual coroutine

<deque>

- `std::deque<T, Allocator>`
- Double-ended queue

`<exception>`

- Base class `std::exception`
- Terminate handler
- `exception_ptr`, shared pointer type for storing any exception object
- Library support for nested exceptions

`<execution>`

(C++17)

- Execution policies for concurrent algorithms
- `seq, par, par_unseq, unseq`

`<filesystem>`

(C++17)

- Access and manipulate files, directories and the pathes that identify them

`<format>`

(C++20)

- Text formatting library
- Type safe and extensible
- `std::format("{} + {} = {}", 5, 6, 11)`
- `template<> struct formatter<MyType>`

<forward_list>

(C++11)

- `std::forward_list<T, Allocator>`
- Singly-linked list

`<fstream>`

- File I/O implementation
- `filebuf`
- `ifstream`
- `ofstream`
- `fstream`

`<functional>`

- Function objects (e.g. `plus, minus, less`)
- Standard hash function
- `function<Signature>`
- `reference_wrapper<T>`
- `invoke`

`<future>`

(C++11)

- Types and functions related to asynchronous tasks
- `future`, `shared_future`, `promise`, `packaged_task`
- `async`

`<initializer_list>`

(C++11)

- `std::initializer_list<T>`
- A view over a constant array

`<iomanip>`

- Stream manipulators that are invoked with arguments (e.g. `setbase(16), setfill('*')`)
- `quoted`, I/O function to read or write a quoted string (C++14)

`<ios>`

- Stream manipulators that are invoked without arguments (e.g. `boolalpha, skipws`)
- `ios_base`, base class for all I/O streams
- `std::ios`, base class to interfer with streambufs

`<iosfwd>`

- Forward declarations for the I/O library

<iostream>

- cout
- cerr
- clog
- cin

<istream>

- std::istream
- std::iostream

`<iterator>`

- Tag types and concepts for the six kinds of iterators
- Iterator traits
- Adaptors and utility functions

`<latch>`

(C++20)

- Single-use thread barrier

`<limits>`

- `numeric_limits<T>`
- Query properties of fundamental numeric types

`<list>`

- `std::list<T, Allocator>`
- Linked list

`<locale>`

- `std::locale`
- Access and manipulate cultural differences like character classification and number formatting

`<map>`

- Sorted maps of key-value pairs
- `std::map<Key, T, Compare, Alloc>`
- Enforces unique keys
- `std::multimap<Key, T, Compare, Alloc>`
- Possibly non-unique keys

`<memory>`

- Dynamic memory management
- Smart pointers (`unique_ptr`, `shared_ptr`, `weak_ptr`)
- Atomic `shared_ptr` and `weak_ptr`
- Traits for allocators and pointer-like types
- Utility functions to work with raw storage

`<memory_resource>`

(C++17)

- `std::pmr::polymorphic_allocator`, an allocator that wraps a polymorphic `std::pmr::memory_resource`
- Three predefined memory resources: `synchronized_pool_resource`, `unsynchronized_pool_resource`, `monotonic_buffer_resource`

`<mutex>`

(C++11)

- Synchronization support by mutual exclusive access
- `mutex`, `timed_mutex`, `recursive_mutex`, `recursive_timed_mutex`
- RAII locking via `lock_guard`, `unique_lock`, `scoped_lock`
- Variadic locking functions `try_lock`, `lock`

<new>

- Low level memory management
- `new_handler`
- Overloads of `operator new` & `operator delete`
- Including overloads taking `nothrow_t`, `align_val_t` & `destroying_delete_t`

`<numbers>`

(C++20)

- Provides mathematical constants
- `std::numbers::pi`

`<numeric>`

- Provides numeric algorithms
- `iota`, `accumulate`, `inner_product`, ...
- `gcd`, `lcm` (C++17)
- `midpoint` (C++20)

`<optional>`

(C++17)

- `std::optional<T>`
- May or may not hold a value of type `T`
- `nullopt_t`

`<ostream>`

- `std::ostream`
- Output Manipulators (e.g. `endl, ends, flush`)

<queue>

- `std::queue<T, Container>`
- A container adaptor that represents a FIFO queue
- `std::priority_queue<T, Container, Comp>`
- A container adaptor that represents a heap

`<random>`

(C++11)

- Generate random numbers
- Randomness comes from Uniform Random Bit Engines (URBGs)
- Based on this, numbers are generated by distributions

`<ranges>`

(C++20)

- Concepts (e.g. `range`, `borrowed_range`, `view`)
- Factories (e.g. `views::single`, `views::iota`)
- Adaptors (e.g. `views::filter`, `views::reverse`)

`<ratio>`

(C++11)

- `std::ratio<Num, Den>`
- Compile-time rational arithmetic and comparison
- Ratio typedefs (e.g. `micro`, `centi`, `giga`)

<regex>

(C++11)

- Regular expression library

`<scoped_allocator>`

(C++11)

- `scoped_allocator_adaptor<OuterAlloc, InnerAlloc...>`
- Multi-Level allocator with multi-level containers

`<semaphore>`

(C++20)

- `counting_semaphore<LeastMaxValue>`
- Synchronization primitive that controls access to a shared resource
- An internal counter gets initialized in the constructor and counts the access

`<set>`

- Sorted sets of keys
- `std::set<Key, Compare, Alloc>`
- Enforces unique keys
- `std::multiset<Key, Compare, Alloc>`
- Possibly non-unique keys

`<shared_mutex>`

(C++14)

- Shared and mutual exclusive access
- `std::shared_mutex` (C++17)
- `std::shared_timed_mutex`
- `std::shared_lock`

`<source_location>`

(C++20)

- `std::source_location`
- Describes a file name, function name, line and column number in the source code
- `std::source_location::current()` gets evaluated at the call site

`<span>`

(C++20)

- `std::span<T, Extent>`
- Non-owning view over a contiguous sequence
- `dynamic_extent`

`<sstream>`

- String streams
- `stringbuf`
- `istringstream`
- `ostringstream`
- `stringstream`

<stack>

- `std::stack<T, Container>`
- A container adaptor that represents a LIFO stack

`<stdexcept>`

- Defines common exception types
- `runtime_error`
- `invalid_argument`
- `out_of_range`
- `...`

`<stop_token>`

(C++20)

- Provide thread cancellation
- `std::stop_source`
- `std::stop_token`
- `std::stop_callback`

`<streambuf>`

- `std::streambuf`
- Controls input and output to a character sequence

`<string>`

- `std::string`
- Stores and manipulates a sequence of characters
- Typedefs for all character types `char`, `char8_t` `char16_t`, `char32_t`, `wchar_t`
- `char_traits<CharT>`

`<string_view>`

(C++17)

- `std::string_view`
- A read-only string view
- Typedefs for all character types `char`, `char8_t` `char16_t`, `char32_t`, `wchar_t`

`<syncstream>`

(C++20)

- `std::osyncstream`, synchronized `ostream` wrapper
- Like a lock_guard for concurrent access to a streams or streambufs destination buffer

`<system_error>`

(C++11)

- Provides low level error codes
- `std::system_error`, exception type containing an `error_code`

`<thread>`

(C++11)

- `std::thread`
- `std::jthread` (C++20)
- Namespace `std::this_thread` to access current thread

<tuple>

(C++11)

- std::tuple<Ts...>
- make_tuple, tie, forward_as_tuple, tuple_cat, apply, make_from_tuple

`<typeindex>`

(C++11)

- `std::type_index`
- A wrapper around `std::type_info`
- Provides a hash as well as being copyable

`<typeinfo>`

- `std::type_info`, as a result of calling `typeid(...)`
- Stores runtime type information (RTTI) of a type

`<type_traits>`

(C++11)

- Metaprogramming facilities to inspect and transform types
- `is_pointer, is_rvalue_reference`
- `is_nothrow_move_constructible`
- `add_const remove_cvref`

`<unordered_map>`

(C++11)

- Hash maps of key-value pairs
- `std::unordered_map<Key, T, Hash, KeyEqual, Alloc>`
- Enforces unique keys
- `std::unordered_multimap<Key, T, Hash, KeyEqual, Alloc>`
- Possibly non-unique keys

`<unordered_set>`

(C++11)

- Hash sets of keys
- `std::unordered_set<Key, Hash, KeyEqual, Alloc>`
- Enforces unique keys
- `std::unordered_multiset<Key, Hash, KeyEqual, Alloc>`
- Possibly non-unique keys

<utility>

- pair<T, U>
- integer_sequence<T, T... Ints>
- General utility functions
- move, forward, as_const
- swap, exchange
- cmp_equal, *_not_equal, *_less, ...
- in_range

`<valarray>`

- `std::valarray<T>`
- Numeric vector

`<variant>`

(C++17)

- `std::variant<Ts...>`
- Stores a value of one the types specified
- `monostate`

`<vector>`

- `std::vector<T, Allocator>`
- Dynamic contiguous array

`<version>`

(C++20)

- Library feature-test macros
- `__cpp_lib_unreachable`

75 C++ STL HEADERS

<algorithm> <any> <array> <atomic> <barrier> <bit> <bitset> <charconv> <chrono>

<compare> <complex> <concepts> <condition_variable> <coroutine> <deque> <exception>

<execution> <filesystem> <format> <forward_list> <fstream> <functional> <future>

<initializer_list> <iomanip> <ios> <iosfwd> <iostream> <istream> <iterator> <latch>

<limits> <list> <locale> <map> <memory> <memory_resource> <mutex> <new> <numbers>

<numeric> <optional> <ostream> <queue> <random> <ranges> <ratio> <regex>

<scoped_allocator> <semaphore> <set> <shared_mutex> <source_location> <sstream>

<stack> <stdexcept> <stop_token> <streambuf> <string> <string_view> <strstream>

<syncstream> <system_error> <thread> <tuple> <typeindex> <typeinfo> <type_traits>

<unordered_map> <unordered_set> <utility> <valarray> <variant> <vector> <version>