C++20
St. Louis C/C++ Meetup

Adam Mitz • mitza@objectcomputing.com • 3.13.2019

C++ Standard Revisions

Starting with the 2011 revision "C++11," the ISO C++ committee produces a new standard every 3 years:

- **Shipped: C++11, C++14, C++17**
- **In work: C++20 (AKA C++2a until finalized)**

Cmte meeting last month was the "feature freeze" for C++20

"C++20 may well be as big a release as C++11" -- someone on Reddit

Technical Specs (TS) Merged to C++20

- Concepts
- Concurrency (partial?)
- Ranges
- Modules
  - Stdlib is not modularized in this version
- Coroutines
  - Limited (no?) stdlib support, see cppcoro library (non-std)

Individual features from these TSs are listed on the following slides, along with other C++20 features.
This list is most likely incomplete and is subject to change as C++20 is finalized.

Standard Library Features

- endian
- remove_cvref
- make_shared for arrays, bounded arr traits
- osyncstream
- string{,_view}: starts_with, ends_with
- to_address
- span, ssize
- Consistent container erasure
  - erase(container, value)
  - erase_if(container, predicate)
- ispow2, ceil2, floor2, log2p1
- bit_cast
- polymorphic_allocator<>
- midpoint, lerp

- **atomic**_ref
- **Atomic** shared_ptr / weak_ptr
- Floating point **atomic**

- **constexpr** <algorithm> and <utility>
- More **constexpr** <complex>

- Precalculated hash values
- execution::unsequenced_policy
- Calendar and timezone, chrono I/O
- <compare> for operator<=>
- <version>
- Concepts library <concepts>
- Ranges library <ranges>

Language Features (1 of 2)

- **Lambda** capture [=, this]
- Template param list in **lambdas**
- **Lambdas** in unevaluated contexts
- Simplify implicit **lambda** capture
- Default construct/assign stateless **lambdas**
- Pack expansion in **lambda** init-capture

- Designated **initializers**
- Default member **init** for bit-fields
- **Init** in range-based for
- **Init** list constructors in CTAD[1]

[1]C++17 Class Template Argument Deduction

- Fixed **const** mismatch in default copy ctor
- **const**&-qualified pointer-to-member rvalue

- Bypass access check for **specializations**
- ADL and function **templates** not visible
- Make **typename** more optional
- Class types in non-type **template** params

- **Attributes** [[likely]] and [[unlikely]]
- **Attribute** [[no_unique_address]]

- Destroying operator delete
- explicit(bool)

Language Features (2 of 2)

- **constexpr** virtual functions
- Specify when **constexpr** member functions are defined
- is_**constant**_evaluated, consteval
- **constexpr** try-catch
- **constexpr** dynamic_cast & typeid
- **constexpr** union (changing active member)

- Prohibit **aggregates** with user-declared constructors
- Parenthesized init of **aggregates**

- Structured binding extensions
  - Lambda capture, static/thread_local
- Stronger Unicode requirements
- Feature-test macros
- Immediate functions
- Nested inline namespaces
- char8_t (distinct type of unsigned char)
- __VA_OPT__ (optional parts of PP varargs)
- Three-way comparison <=>

- Concepts
- Contracts
- Modules
- Coroutines

Approved for C++20 but not yet in draft

- Expansion statements
- C++20 Sync Lib:
  - atomic_wait / notify
  - atomic_flag_*
  - atomic_{un,}signed_lock_free
  - Semaphores, latches, barriers
- <format>
- Conceptification of iterators
- std::optional "monadic" (chained) ops
- source_location, Stack Trace lib
- flatmap, flatset
- unique_function (move-only std::function)
- ostream_joiner
- byteswap

- **constexpr** allocations (limited), std::vector
- **constexpr** type_info ==
- **constexpr** <cmath>, <cstdlib>

- Deprecate some uses of volatile
- using enum
- Removing UB of using untyped storage as objects

References

https://isocpp.org/std/status

https://en.cppreference.com/w/cpp/compiler_support#cpp2a

https://www.bfilipek.com/2019/02/papers-kona.html

https://cor3ntin.github.io/posts/kona2019

https://www.reddit.com/r/cpp/comments/au0c4x/201902_kona_iso_c_committee_trip_report_c20

https://github.com/lewissbaker/cppcoro