C++23 Features Reference Card

Language Features

New or updated keywords and preprocessor directives:

- ✔ `auto` - Explicit cast to a prvalue copy
- ✔ `this` – explicit object parameter
- ✔ `if consteval`
- ✔ `#elifdef` / `#elifndef` / `#warning` - for C23 compatibility

Deducing `this`, P0847, CRTP Example

A way to explicitly pass "this" parameter into a member function, gives more control and reduce code. Useful for passing `this` by value, recursive lambdas, CRTP and more.

```
struct Pattern {
  template <typename Self> void foo(this Self&& self)
    { self.fooImpl(); } };
```

Extend init-statement to allow alias-declaration, P2360

In C++20 `using` wasn't allowed in the for loop, now it's fixed:

```
for (using T = int; T e : container) { ... }
```

Multidimensional subscript operator, P2128

Change the rules to allow multiple params into the operator[]:

```
struct Array2D {int operator[](size_t i, size_t j)
{ return m[i+j*w];} };
```

This is crucial for types like `std::mdspan`

Features for Lambdas

- ✔ Attributes on lambdas,
- ✔ `()` is more optional,
- ✔ the call operator can be `static`,
- ✔ deducing `this` adds new capabilities like better recursion.
- ✔ Change scope of lambda trailing-return-type

`[[assume]]` new attribute P1774

`[[assume]]` Specifies that an expression will always evaluate to true at a given point. It standardizes the existing vendor specific semantics like __builtin_assume (Clang) and __assume (MSVC, ICC). Offers potential optimization opportunities for compilers.

`constexpr` Updates P2448, P2647 and P2242

- ✔ Relax rules for constructors and return types for `constexpr` functions, making it almost identical to regular functions.
- ✔ Permitting static constexpr variables in constexpr functions

Extend Lifetime of Temporaries in range based for, P2718

The lifetime of temporary objects in the for-range-initializer is extended until the end of the loop. C++20 code below generates UB as the temporary object is destroyed when the function returns.

```
std::vector<std::vector<int>> getVector();
for (auto e : getVector()[0])
```

Library Features

`std::generator` coroutine generator, P2502

A library support for a synchronous generator. It models view and input_range of the elements yielded by the evaluation of a co-routine.

Stacktrace library, P0881

Based on Boost.Stacktrace, allows to get more context when debugging code. The library defines components to store the stacktrace of the current thread of execution and query information about the stored stacktrace at runtime.

```
std::cout << std::stacktrace::current();
```

std::is_scoped_enum and std::to_underlying

A library support for enums and underlying types. Equivalent to:

```
static_cast<std::underlying_type_t<Enum>>(e);
```

the trait (`underlying_type`) has been available since C++11.

std::string/std::string_view improvements:

- ✔ `contains(char/string_view/const char*)` - member fun
- ✔ Prohibiting `std::basic_string` and `std::basic_string_view` construction from `nullptr`
- ✔ Range constructor for `std::basic_string_view`
- ✔ `string::resize_and_overwrite()` - Allows us to initialize/resize strings without clearing the buffer but filling bytes with some user operation.

std::out_ptr(), std::inout_ptr(), P1132

Functions that wrap a smart pointer into a special type allowing to pass to low level functions that require pointer to pointer parameters.

Ranges and Views additions

- ✔ **ranges::to<>, P1206 – build a collection from a range**
- ✔ `ranges::starts_with()` and `ranges::ends_with()`
- ✔ `ranges::iota()`, `ranges::shift_left/right()`
- ✔ `ranges::find_last(_if)()`, `find_last_if_not()`
- ✔ `ranges::contains()` and `ranges::contains_subrange()`
- ✔ Ranges fold algorithms: `ranges::fold_*()`
- ✔ Views: `slide, cartesian_product, repeat, enumerate, adjacent(_transform), as_rvalue, as_const, stride, chunk(_by), join_with, zip(_transform)`

Heterogeneous erasure for associative containers, P2077

Continuation of the work for heterogeneous operations. This time you can use transparent comparators for `erase()` and `extract()` member functions. To be backward compatible the comparators cannot be convertible to `iterator` or `const_iterator` of a given container.

Monadic operations for `std::optional`, P0798

New member functions for optional: and_then, transform and or_else.

```
auto ret = userName.transform(toUpper)
.and_then([](string x) { ... }
.or_else...
```

<expected> and monadic operations, P0323

A vocabulary type that allows storing either of two values: T or unexpected (in a form of some error Type).

```
Enum FuelErr { … }
std::expected<double, FuelErr> calcFuel(int dst);
```

`<flat_map>` and `<flat_set>`, P0429 and P1222

Drop-in replacement for maps and sets with better performance characteristics.

`std::mdspan` – multidimensional span, P0009

A generalization over `std::span` for multiple dimensions. Supports dynamic as well as static extents (compile time constants). It also supports various mappings like column-major order, row-major or even stride access.

Formatted output library `<print>`, P2093

```
std::print("{1} {0}!\n", "World", "Hello");
```

New functions in the `<print>` header: `std::print`, `std::println` (adds a new line) that uses `std::format` to output text to `stdout`. Plus lower-level routines like `vprint_unicode` with more parameters for output.

Standard Library Modules, P2465

`import std;` imports everything in `namespace std` from C++ headers and C wrapper headers. It also imports ::operator new etc. from <new>.

`import std.compat;` imports all of the above, plus the global namespace counterparts for the C wrapper headers.

<spanstream>: string-stream with span buffers P0448

New classes: `basic_spanbuf`, `basic_ispanstream`, `basic_ospanstream`, `basic_spanstream` analogous to existing stream classes but using `std::span` as the buffer. They allows explicit buffer management and improved performance.

Other (Language & Library)

- ✔ If consteval { }, P1938
- ✔ auto(x) and auto{x}, P0849 - eplaces decay_copy
- ✔ static operator() and static operator []
- ✔ CTAD from inherited constructors
- ✔ Unicode improvements
- ✔ `std::visit()` for classes derived from `std::variant`
- ✔ `std::unreachable()`
- ✔ Deprecating `std::aligned_storage` and `aligned_union`
- ✔ Simpler implicit move P2266
- ✔ Pipe support for user-defined range adaptors, P2387
- ✔ std::format improvements – formatting ranges, compile time parsing and more
- ✔ constexpr std::unique_ptr, P2273
- ✔ constexpr to_chars() and from_chars() for integers, P2291
- ✔ Explicit lifetime management, P2590
- ✔ Literal suffix for (signed) size_t – uz, UZ, P0330

References

isocpp.org,
en.cppreference.com/w/cpp/compiler_support
www.cppstories.com

Get Extended Version of this ref card @CppStories Patreon

 Last update October 2024