C++ 20 Features

Presentation Prepared by Jawad Srour

Table of contents

01 constexpr

Refresher on constexpr specifier and its use-cases

02 consteval

Introducing consteval specifier

03 constinit

Introducing constinit specifier

04 Modules

Introducing modules and import/export modules

05 Spaceship

The new comparison operator

01

constexpr



—Someone Famous

“Everything that can be declared as const, should be declared as const.”

constexpr

- **constexpr** is a feature added in C++ 11.
- The main idea is a performance improvement of programs by doing computations at compile time rather than run time.
- constexpr tells the compiler that the value can be used in other constant expressions
- Functions can be declared as constexpr
- We can make use of it to convert units from one system to another

But note the below:

- function should not be of void return type
- constexpr functions can only call other constexpr functions

constexpr VS const

constexpr



Variables Must be initialized at compile time



We can use functions to evaluate compile time constants



We can create objects at compile time

const



Variables May be initialized at compile time or runtime

02

consteval



consteval

- Specifies that a function is an immediate function
- Immediate function: every call to the function must produce a compile-time constant expression.
- If any declaration of a function or function template contains a consteval specifier, then all declarations of that function or function template must contain that specifier.

consteval VS constexpr

consteval



Evaluation of functions is ensured at compile-time



We can only use it for functions, not variables



Can’t have any side-effects and can’t depend on run-time state

constexpr



Functions might be evaluated at runtime

03

constinit

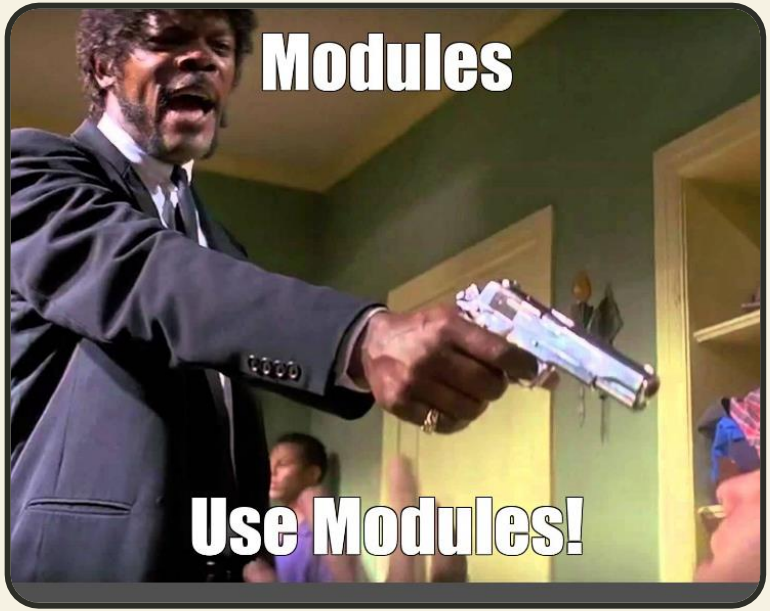


constinit

- Asserts that a variable has static initialization
  - zero initialization
  - constant initialization (ex: string literal)
- Useful when we want to ensure that a variable or object is initialized before any other code is executed
- Useful when we have expensive operations that we prefer to be guaranteed to get executed at compile time

04

Modules



Modules

- Modules are a way to share declarations and definitions across translation units, just like the headers that we are used to.
- Modules are a ore modern and flexible approach to sharing code across translation units.
- It provides the flexibility of exporting only some our functions in an elegant way.
- For those familiar with JavaScript, it works pretty much the same way.
- Uses the file extension `.cppm` which stands for C++ module to indicate a file coontains a module interface unit.

05

Spaceship



Spaceship Operator

• We used to implement all comparison operators to achieve comparison behavior for our custom types

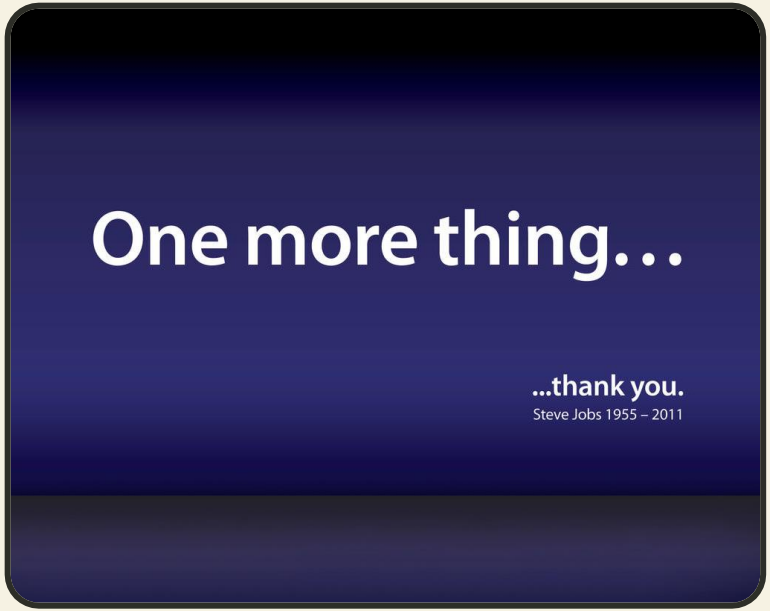
• As of C++20, we can overload == without overloading != … the latter is guaranteed to be generated by the compiler

•However, we now have the three-way comparison operator also known as spaceship operator to further reduce boilerplate code

• We can do a default spaceship operator implementation

- automatically defaults equality operator…
- might be useful for simple types such as Point class

05 Honorable Mentions



This Photo by Unknown Author is licensed under CC BY-SA

std::midpoint and std::lerp

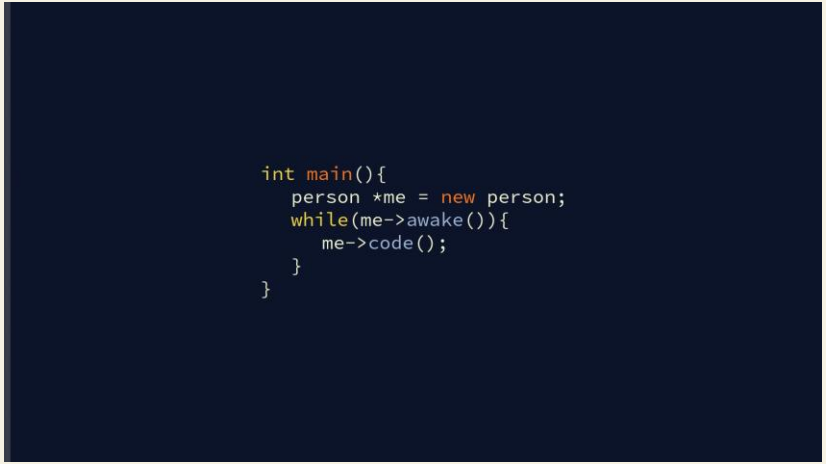
• std::midpoint → computes the midpoint of integers, floating points, or pointers a and b

•std::lerp → computes the linear interpolation between a and b, if the parameter t is inside [0, 1], the linear extrapolation otherwise

- can use std::clamp for ensuring range [0, 1]
- can use std::fabs to ensure positive values (good practice)

Future Proposals

- Time-zone APIs as of C++20
  - Locate_zone(name)
  - Current_zone()
- C++ introduced Calendar functions in C++20 (Defined in the chrono library)
  - https://en.cppreference.com/w/cpp/chrono#Calendar
- C++23 features (flat_map, flat_set,erase_if, print, println, ..)
- std::expected as of C++23 is a very useful feature



```
int main(){
   person *me = new person;
   while(me->awake()){
      me->code();
   }
}
```

Thanks!

Questions?

jsrour@murex.com

Presentation by: Jawad Srour

References

1. https://stackoverflow.com/questions/4748083/when-should-you-use-constexpr-capability-in-c11
2. https://en.cppreference.com/w/cpp/language/constinit