C++ Roadmap

2.5. Advanced output (*)
- #include <format>
- std::format()
- std::print()
- std::println()
- format string
- format specifiers

2.2. Type conversion (*)
- concept of casting
- C-style cast
- static_cast
- const_cast

2.1. In-depth primitive data types (*)
- signedness
- integer types of different size
- ASCII
- char arithmetic
- capitalizing char using math
- std::toupper()
- std::tolower()

1. Basic program
- program structure
- output operation
- #include <iostream>

2.4.1. String searching (*)
- std::string::find()
- std::string::npos

2.3.2. Type deduction (*)
- auto
- deduction based on literals
- declaration by initialization
- auto and string literals
- std::initializer_list
- trailing return type notation

2. Variables
- primitive types
- variables
- input operation
- declaration
- initialization
- const

2.4. Strings
- std::string
- std::cin for std::string
- std::getline() for std::string

2.3. Literals (*)
- char literals
- string literals
- int literals
- float literals
- double literals
- unsigned literals
- various size int literals

3. Basic math
- int vs float
- basic int math
- basic float math
- mixing int and float in math
- modulo division

2.6. Compile-time evaluation (*)
- constexpr
- consteval

2.4.2. String modification (*)
- string concatenation
- std::string::substr()
- std::stringstream

2.3.1. Escape characters (*)
- new line character
- backspace character
- tabulation character
- carriage return character

4. Conditionals
- if statement
- boolean type
- switch case
- preincrementation
- postincrementation

3.1. Advanced floating-point (*)
- IEEE754
- rounding precision
- numeric stability

4.1. Implicit bool conversions (*)
- evaluating numbers as bool
- evaluating characters as bool

5. Iterations
- for loop
- while loop
- do while loop

4.2. Compile-time conditionals (*)
- if constexpr

7.2. Scopes
- local scope
- global scope

6. Static arrays
- raw array
- std::array<>
- ranged for
- std::string as an array

6.1. Advanced static arrays (*)
- C-string
- multidimensional raw arrays
- multidimensional std::array<>

7.1. Advanced functions (*)
- function prototype
- function overloading
- Tail Recursion Optimization (TRO)
- Return Value Optimization (RVO)
- inline functions
- default parameters
- constexpr functions

7. Functions
- making a function
- function body
- parameters
- returning a value
- early return
- recursion

7.3. Side effects (*)
- pure functions
- static variables

8.1.2. Linking errors (*)
- symbol missing in translation unit
- missing library
- duplicated symbols

8.2. Macros (*)
- preprocessor directives
- defines
- conditional compilation

8.3. Modules (*)
- modules
- exporting symbols

8. Splitting code into files
- header files
- source files
- #include in detail
- #pragma once

8.1. In-depth codebase structure (*)
- compilation process
- translation units

8.1.4. Integration with C code (*)
- extern "C"

- conditional compilation
- include guards vs #pragma once

- importing modules

- #pragma once
- function prototypes in header files
- function bodies in source files

- linking
- forward declarations

- extern "C"

9.1. In-depth enums (*)
- enum base type
- assigning values to enum elements
- casting enums
- conversion functions for enums

9. Enumeration types
- enum
- scoped enum

8.1.1. External libraries (*)
- static libraries
- dynamic libraries

8.1.3. Managing compiler optimizations (*)
- volatile
- compiler attributes

10. File operations
- std::fstream
- std::getline with files
- std::stoi, std::stof, std::stod

10.1. Advanced file operations (*)
- overloading stream operator
- std::filesystem
- binary files

11.3. Structure memory concerns (*)
- memory alignment
- padding
- bit fields

11. Structures
- structures
- nesting structures
- brace initialization
- brace initialization with explicit field names

11.5. Unions (*)
- union
- std::variant<>
- undefined behavior of unions

11.4. In-depth look into structures (*)
- access modifiers (public, private)
- struct default access

12.1. Anonymous namespaces (*)
- anonymous namespaces
- file-restricted visibility

11.1. Default values (*)
- default field values
- brace initialization vs default values

12. Namespaces
- namespaces
- name collisions

11.2. Structure data access (*)
- aggregates
- structure methods
- structured binding
- dot operator

13. Access to original data
- raw pointers
- references
- universal references
- name of a raw array as a pointer
- pointer function parameters
- reference function parameters
- arrow operator

13.1. Conversions of pointers (*)
- reinterpret_cast<>
- dynamic_cast<>
- C-style cast

13.2. Advanced pointers (*)
- universal pointer
- pointer to function
- memory ownership
- memory ownership transfer
- pointer arithmetic

14. Dynamic allocation
- std::unique_ptr<>
- std::shared_ptr<>
- std::weak_ptr<>
- std::vector<>
- std::list<>
- new operator
- delete operator
- delete[] operator

14.2. Iterators (*)
- begin() method
- end() method
- iterators
- using iterators in for loop
- iterating over non-contiguous memory

14.3. Advanced memory management (*)
- buffers
- placement new
- cache-friendly programming

14.1. Memory model (*)
- stack
- heap
- contiguous memory
- non-contiguous memory
- memory fragmentation
- random memory access

15. Object Oriented Programming Basics
- classes
- default constructors
- parametric constructors
- destructors
- methods
- fields
- RAII

14.4. Optional values (*)
- std::optional<>
- accessing optional value
- value_or()

15.1. Leveraging compiler method generation (*)
- default keyword
- class method generation rules
- Rule of Zero

16. OOP Data Management Concepts
- encapsulation
- access specifiers (private, public)
- default class access
- getters
- setters

15.2. Class performance optimization (*)
- move semantics
- assignment operator
- move assignment operator
- Rule of Five
- .push_back() vs .emplace_back() in std::vector<>

/

17. Advanced access of class contents
- static members
- static methods
- scope operator
- friend classes
- friend functions

17.2. Memory access to class content (*)
- class member pointer
- class method pointer

17.1. Making class compliant with advanced language features (*)
- structured-binding-compliant interface
- custom iterators
- custom begin() and end() method
- custom cbegin() and cend() method
- custom rbegin() and rend() method

18.1. Advanced object construction (*)
- explicit constructors
- constexpr constructors
- std::initializer_list
- initializer list constructor
- default parameters in constructors
- multiple parametric constructors
- private constructors
- delegating object construction
  to a static method
- construction guards

18. Const-suitable classes
- what const object implies for methods
- const methods

18.2. Advanced const object properties (*)
- volatile members inside const objects

19. Inheritance
- protected access specifier
- public inheritance
- protected inheritance
- private inheritance
- name shadowing
- accessing private parent members
- inheritance also possible with structs
- protected also applying to structs

19.1. Multiple inheritance (*)
- multiple inheritance
- diamond problem
- accessing member of a specific parent

20.2. Advanced operator overloading (*)
- overloading new operator
- overloading delete operator
- overloading delete[] operator
- conversion operators
- functors
- rules for implicit conversion
  of custom types

20. Polymorphism
- what is polymorphism
- virtual functions
- abstract classes
- purely virtual functions
- virtual destructors
- method overloading
- operator overloading
- override
- final

20.3. Improving readability (*)
- type aliases
- namespace aliases
- using type
- using namespace
- decltype

20.1. Advanced polymorphism concepts (*)
- vtable
- dynamic_cast
- rules for downcasting

22.3. Templates vs codebase (*)
- template rules regarding linking
- template forward declarations
- moving class template
  method definitions to source files

20.4. Design Patterns
- creational patterns
- structural patterns
- behavioral patterns
- singleton using static function

21. Error handling
- exceptions
- #include <stdexcept>
- try / catch blocks

21.1. Advanced error handling (*)
- static_assert
- assert
- noexcept
- custom exceptions
- different handling for
  different exceptions

22.2. Adaptive templates (*)
- std::enable_if
- std::declval
- type_traits library
- CRTP - Curriously
  Recurring Template Pattern
- SFINAE - Substitution Failure
  Is Not An Error
- std::decay
- concepts

22. Templates
- teplates
- template full specialization
- template partial specialization
- templates simplified with auto

22.1. Advanced template concepts (*)
- decltype and return type
  based on template parameters
- variadic templates
- parameter pack
- caching using templates

23.2. Popular third-party libraries (*)
- Boost::ASIO
- Qt
- OpenGL
- Vulkan
- CUDA
- PhysX

23. STL - Standard Template Library
- std::pair<>
- std::tuple<>
- std::make_tuple<>
- std::tie
- std::map<>
- std::unordered_map<>
- std::set<>
- std::unordered_set<>
- std::find
- std::ranges::find

23.1. Useful standard libraries (*)
- chrono
- random
- iosfwd

/

**24. Lambdas and
Higher Order Functions**

- lambda
- lambda capture block
- lambda aggregate capture specifiers
- std::find_if
- std::ranges::find_if
- std::ranges::remove_if
- std::ranges::for_each
- algorithm library

24.1. Storing functions (*)

- function pointer
- std::function<>
- callbacks

24.2. Transforming data (*)

- std::views<>
- std::ranges::view<>
- std::transform
- std::filter
- lazy evaluation

25.1. In-depth concurrency

- thread priority
- scheduler
- preemption
- thread blocking
- thread yielding

25. Concurrency

- threads
- parallelism
- concurrency

25.2. Asynchronous programming

- asynchronous operations
- coroutines
- ISR - Interrupt Service
  Routine

**26. Sharing resources
between threads**

- critical section
- semaphore
- mutex
- scoped lock
- blocking queue
- polling

26.1. Synchronized types (*)

- atomic instructions
- #include <atomic>
- atomic types

27.1. Interacting with the OS

- main function parameters
- signal handler
- system calls
- thread vs process

**27. Resolving problems
with concurrency**

- deadlock
- thread starvation
- race condition

/