From Introductory to Advanced C++

Learning Guidelines

Slobodan Dmitrovic

About me

Slobodan Dmitrovic

- C++ Trainer
- R&D Developer
- Author
- linkedin.com/in/slobodan-dmitrovic/

Starting with C++

When starting with C++, first, we need to learn about the following topics in the following order:

- The C++ **language** basics
- The **C++ Standard Library** basics
- Modern C++ **standards**

Establishing a solid knowledge-base is essential in the beginning.

The C++ Language Basics

- Types
- Declarations
- Operands
- Operators
- Expressions
- Statements
- Functions
- Classes
- Templates Basics (only a brief introduction is advised in the beginning)
- More, much more…

The C++ Standard Library basics

The C++ Standard Library **basics** include learning about the widely used:

- Containers
- Algorithms

Modern C++ standards

- Modern C++ standards are C++11, C++14, C++17, C++20 and C++23 standards.
- Initially, we should concentrate on the **notable features** of **C++11**, **C++14**, and **C++17** standards, as those are the most widely used standards in the industry.

What should we learn in the beginning?

- Built-in types
- Declaration
- Definition
- Initialization
- Operands
- Operators
- Expressions
- Statements
- Built-in statements
- Scope and lifetime
- Automatic and dynamic storage
- References
- Functions
- more…

What should we learn when it comes to C++ functions?

- Function declarations
- Function definitions
- Invoking a function
- Passing arguments by value, reference, and const-reference
- The return statement
- Function overloads

How do we start learning about C++ classes?

- Data members
- Member functions
- An instance of a class
- Visibility specifiers
- Constructors
- Member initializer lists
- Destructors

Learning about classes – next steps

- Copy and move semantics in C++
- Operator overloading
- Inheritance
- Virtual and overridden member functions
- Polymorphism
- Interfaces

How do we start learning about C++ templates?

- Basic function template declaration
- Basic class template declaration
- Template specialization

The right amount of template introduction

In the beginning, only a **brief introduction** to C++ template programming is advised.

How do we learn the C++ Standard Library?

When starting with the C++ Standard Library, learn about the following topics:

- Widely used **containers** (std::vector, std::array, std::set, std::map, std::list…)
- **Iterators**
- Widely used **functions** (std::find, std::find_if, std::sort, std::count, std::count_if…)

We do not have to learn the entire standard library by heart. Initially, we need to establish a solid knowledge base.

Learning about different C++ standards

Once we are familiar with the C++ language and C++ Standard Library basics, we should learn about the **notable features** introduced in a several C++ standards. In the beginning, start with the following standards:

- **C++11**
- C++14
- C++17

What to learn next?

Once we learn the C++ basics, what to learn next? What are the topics that make up **intermediate and advanced C++**?

- C++ idioms
- Concurrency through multithreading
- Runtime polymorphism
- In-depth template metaprogramming
- Guidelines, dos and don’ts, avoiding pitfalls
- Software design using C++

C++ idioms

Learn about some of the typical C++ idioms, such as:

- RAII
- The erase-remove idiom
- PIMPL
- Other idioms (that best suit your use-case)

Concurrency through multithreading

The C++ Standard Library offers support for multithreading, mutexes, locking mechanisms, and other related features. Explore:

- std::thread
- std::lock_guard
- std::mutex
- Future and promise wrappers
- other facilities…

Polymorphism

- Learn about virtual and overridden member functions
- Using the std::unique_ptr to achieve runtime polymorphism
- Creating and using interfaces

Also, explore the following topics:

- Curiously Recurring Template Pattern
- std::variant

In-depth template metaprogramming

Once you are familiar with the template basics, learn about:

- Template inheritance
- Specialization
- Explicit instantiation
- Other topics

Guidelines, dos and don’ts, avoiding pitfalls

- Learn about some of the C++-specific guidelines, best practices, and common pitfalls to avoid.
- Please note that you don't have to know all the guidelines by heart, but you should be aware of their existence. Many of them are already implemented in static code analyzers.

Software design using C++

If you're interested in software architecture and design, learn about the:

- SOLID and other principles
- Design patterns in general

This helps when aiming for:

- An elegant and maintainable framework
- The separation of concerns

Thank you!

Q & A