How to approach learning and teaching C++?

by Slobodan Dmitrovic

About this talk

Part I - How to approach learning C++?

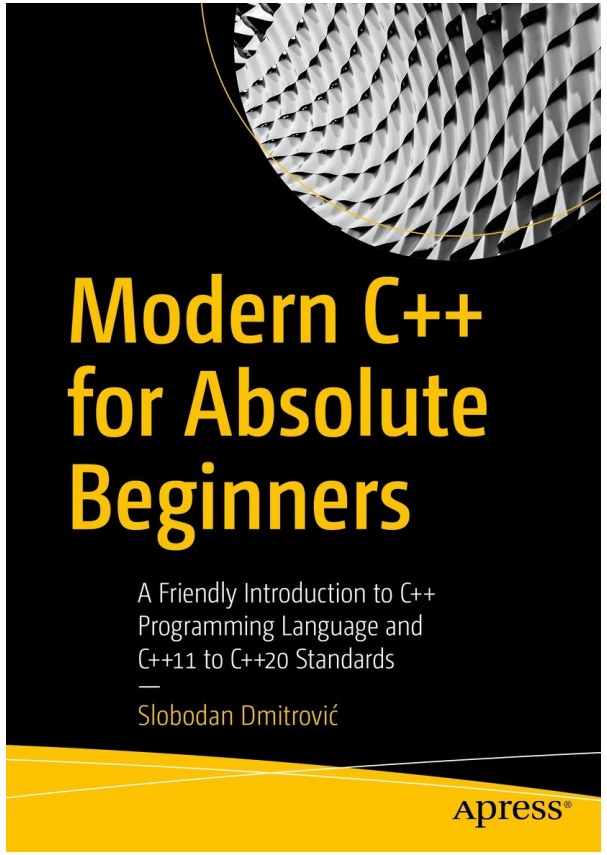
Here we discuss how to approach learning C++ from a C++ beginner's perspective.

Part II - How to approach teaching C++?

Here we discuss how to approach teaching C++ from a C++ trainer's perspective.

About Slobodan Dmitrovic

- C++ trainer, software developer and consultant
- Author of "Modern C++ for Absolute Beginners"
- Slobodan specializes in providing C++ training courses for teams
- info@cppandfriends.com



Part I – How to approach learning C++?

What is C++?

C++ is a programming language.

C++ is a multi-paradigm, object-oriented, systems programming, standardized programming language.

What is there to learn?

- **The C++ language itself**
- **The C++ Standard Library**
- **Modern C++ standards**
- More, much more…

Start with these three things, in this particular order.

The C++ language

Can be roughly divided into three main categories:

- Basic language facilities
- Classes
- Templates

The C++ Standard Library

A library of useful containers and functions we can use in our program.
Every C++ compiler is accompanied by the C++ Standard Library.

C++ standards

There are multiple ISO C++ standards, listed here by their informal names in chronological order:

- C++98
- C++03
- **C++11**
- **C++14**
- **C++17**
- C++20*

Every C++ standard starting with the C++11 is informally referred to as "Modern C++."

Where to learn the C++ from?

Available resources:

- **Books**
- Online courses
- Online videos
- **Training with C++ trainers**
- Blogs and other online resources

Get a couple of books, do the research, opt for at least three introductory books. It is OK to discard one of them as you go along.

C++ trainers can help you achieve in days what would have otherwise taken you months.

C++ is not C with classes

Prefer resources that teach you these:

```
#include <iostream>

int main()
{
    std::cout << "Hello World!" << '\n';
}

//-----------------------------------
std::string s = "Hello World";
//-----------------------------------
class MyClass
{
};
```

To resources that teach you these:

```
#include <cstdio>

int main(void)
{
    printf("Hello World\n");
}

//-----------------------------------
const char* s = "Hello World";
char s2[] = "Hello World";
//-----------------------------------
typedef struct MyClass {
} TMyClass;
```

How not to approach learning C++?

By guessing

- Do not try to learn C++ by guessing
- While some languages can be learned by playing a guessing game, C++ can not

By drawing parallels between C++ and other languages

- Try not to draw parallels between C++ and other languages, C++ is in a league of its own. What works in other languages does not necessarily translate to C++
- C++ is not "C with classes", nor a "subset of Java"

C++ knowledge backbone

Basic facilities

Types and Modifiers

Declaration, Definition, and Initialization

Operators, Expressions

Standard Input and Output

Arrays

Pointers

References

Strings

Automatic Type Deduction

Built-in Statements

Constants

Functions

Storage, Scope, Visibility

Headers and Namespaces

Conversions

Enumerations

Lambdas and range-based loops

More…

Classes and Templates

Data Member Fields

Member Functions

Access Specifiers

Constructors

Default Constructor

Member Initialization

Copy Constructor

Copy Assignment

Move Constructor

Move Assignment

Operator Overloading

Destructors

Inheritance

Polymorphism

Introduction to Templates

More…

The C++ Standard Library

Containers

- std::vector
- std::array
- std::set
- std::map
- …

Iterators

Algorithms and Utilities

- std::sort
- std::find
- std::copy
- Min and Max
- …

More…

Modern C++ Standards Features

From C++11 to C++17 (or C++20?)

How to tackle the complexity?

- **We need to build a solid base first**
- We do not need to go into every detail
- **We do not need to know everything, and that is just fine**
- Learn a subset that fits your use-case
- The Standard Library is there for our own convenience, we do not need to know the entire Standard Library by heart
- Just because the language is complex, we do not need to make it complicated

Why you should learn C++?

- **It is an immensely powerful language**
- Programming in C++ can be an extremely rewarding experience
- C++ is widely used, it covers a lot of domains
- It gets you places, pays well
- A constant source of learning
- C++ developers are in **high demand**

"When I started with C++, I almost lost all interest in other languages…"

"It is an R&D engineer's paradise…"

Part II – How to approach teaching C++?

What to teach to a beginner in C++?

- The C++ programming language
- The C++ Standard Library
- Modern C++ standards

} Basics at first

Try not to overwhelm the trainee with topics such as design patterns, and too many language guidelines. Build a solid base first.

Show the trainee the power of the language. Deliver the relevant, precise and concise information, avoid going into too many details and insisting on border cases.

What not to teach to a beginner in C++?

- How to consume OS-specific interfaces
- The use of 3rd party libraries
- Design patterns
- Graphics
- Sounds
- Network

We want to teach platform-agnostic, portable C++, not domain-specific use-cases.

The structured approach

- How *topic x* relates to *topic y,* which one should you teach first*?*
- Keep the theoretical part at a minimum but do not compromise with the factual content nor the terminology
- Accompany the theoretical introduction with plenty of examples
- Make *example 1* a very basic one
- Make *example 2* more complex, but make sure it expands on *example 1* if possible
- Break the complex topics into smaller pieces and start over, repeat the process with examples
- Deliver the relevant information
- Keep in mind that you are teaching beginners, not other trainers

In a nutshell

- Decide on the topics and the scope
- Decide in which order to teach the topics
- Provide just enough theoretical introduction
- Provide plenty of (generic) examples that increase in complexity
- Do not compromise with the terminology
- Avoid forward referencing as much as possible
- Decide on how much information is too much information
- Deliver the facts, the value
- Please have in mind that it is a constant balancing act

Some challenges and possible solutions

- In modern C++, the use of raw arrays and raw pointers is largely discouraged, do you teach these?
- std::string is not part of the basic language facilities per se, do you teach it there or when teaching about the C++ standard library?
- Which guidelines do you teach at the beginning?

Possible solutions

- Teach raw arrays and pointers if only to discourage their use in favor of std::vector, std::array and smart pointers. It is likely we will still encounter those in everyday use
- The std::string is so integral to a language and everyday operation, so it is fine to teach it when discussing basic language facilities
- The most important, widely used ones

Thank you!

Climbing mountain C++ is both a challenging and rewarding task. But once at the top, the view is breathtaking. I strongly encourage you to take this journey.