Basics Programs in C++

Cristo Daniel Alvarado

January 27, 2026

Contents

Code List

Chapter 1

Basic Codes

1.1 About C++

Citing the book *A tour of C++* by Bjarne Stroustrup:

> *C++ is a compiled language. For a program to run, its source text has to be processed by a compiler, producing object files, which are combined by a linker yielding an executable program.*

Let's analyze this sentence. One of the questions that arises from this is the following: what is a **compiled language**?

Definition 1.1.1 (**Compiled Language**)
A **compiled language** is one where *the program, once compiled, is expressed in the instructions of the target machine.*

For example, an addition "`+`" operation in your source code could be translated directly to the "`ADD`" instruction in machine code.

There are other kind of languages, called **interpreted languages**:

Definition 1.1.2 (**Interpreted Language**)
An **interpreted language** is one where *the instructions are not directly executed by the target machine, but instead read and executed by some other program* (which normally is written in the language of the native machine).

For example, the same "`+`" operation would be recognised by the interpreter at run time, which would then call its own "`add(a,b)`" function with the appropriate arguments, which would then execute the machine code "`ADD`" instruction.

Now, what is an **object file**?

Definition 1.1.3 (**Object File**)
An **object file** is *the real output from the compilation phase.*

It's mostly machine code, but has info that allows a linker to see what symbols are in it as well as symbols it requires in order to work.

And, what is a linker?

Definition 1.1.4 (**Linker**)
A **linker** or **link editor** is a *computer program that combines intermediate software build files such as object and library files into a single executable file such as a program or library.*

So, for example, a C++ program consist of many source file codes (sometimes simply called **source files**).

The process is as follows:

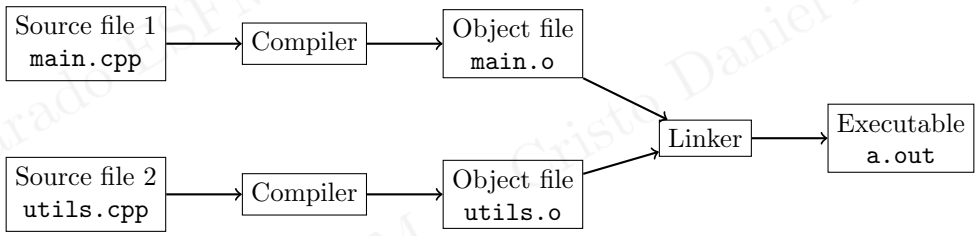

Figure 1.1: C++ compile process.

Observation 1.1.1 (**Portability of Source Code**)
An executable program is created for a specific hardware/system combination; it is not portable, say, from a Mac to a Windows PC.

When we talk about portability of C++ programs, we usually mean **portability of source code**; that is, the source code can be successfully compiled and run on a variety of systems.

Definition 1.1.5 (**Entities in the ISO C++**)
The ISO C++ standard defines two kinds of entities:

- **Core language features**, such as built-in types (e.g., `char` and `int`) and loops (e.g., `for`-statements and `while`-statements)
- **Standard-library components**, such as containers (e.g., `vector` and `map`) and I/O operations (e.g., `<<` and `getline()`)

Chapter 2

Object Oriented Programming

2.1 OOP

Definition 2.1.1 (**OOP Object Oriented Programming**)
OOP stands for **Object-Oriented Programming**. Object-oriented programming is about *creating objects, which can hold data and functions that work on that data.*