C++ Programming Basics

Difficulty Level : Easy • Last Updated : 11 Mar, 2023

C++ is a general-purpose programming language and is widely used nowadays for competitive programming. It has imperative, object-oriented, and generic programming features. C++ runs on lots of platforms like Windows, Linux, Unix, Mac, etc.

Before explaining the basics of C++, we would like to clarify two more ideas: **low-level** and **high-level**. To make it easy to understand, let's consider this scenario – when we go to the Google search engine and search for some queries, Google displays some websites according to our question. Google does this for us at a very high level. We don't know what's happening at the low level until we look into Google servers (at a low level) and further to the level where the data is in the form of 0s/1s. The point we want to make here is that a low level means nearest to the hardware, and a high level means farther from the hardware with a lot of layers of abstraction. **C ++ is considered a low-level language** as it is closer to hardware than most general-purpose programming languages. However to become proficient in any programming language, one Firstly needs to understand the basics of that language.

Basics of C++ Programming

1. Basic Syntax and First Program in C++

Learning C++ programming can be simplified into writing your program in a text editor and saving it with the correct extension(.CPP, C, CP), and compiling your program using a compiler or online IDE. The "Hello World" program is the first step toward learning any programming language and is also one of the simplest programs you will learn.

We can learn more about C++ Basic Syntax here – C++ Basic Syntax

2. Basic Input and Output in C++

C++ comes with libraries that provide us with many ways for performing input and output. In C++ input and output are performed in the form of a sequence of bytes or more commonly known as streams. The two methods **cin** and **cout** are used very often for taking inputs and printing outputs respectively. These two are the most basic methods of taking input and output in C++.

To learn more about basic input and output in C++, refer to this article – Basic Input/Output in C++

3. Comments in C++

A well-documented program is a good practice for a programmer. It makes a program more readable and error finding becomes easier. One important part of good documentation is Comments. In computer programming, a comment is a programmer-readable explanation or annotation in the source code of a computer program. These are statements that are not executed by the compiler and interpreter.

We can learn more about C++ comments in this article – C++ Comments

4. Data Types and Modifiers in C++

All variables use data type during declaration to restrict the type of data to be stored. Therefore, we can say that data types are used to tell the variables the type of data they can store. Whenever a variable is defined in C++, the compiler allocates some memory for that variable based on the data type with which it is declared. Every data type requires a different amount of memory. We can change this by using data type modifiers.

To know more about data types, refer to this article – C++ Data Types
To know more about type modifiers, refer to this article – C++ Type Modifiers

5. Variables in C++

A variable is a name given to a memory location. It is the basic unit of storage in a program. The value stored in a variable can be changed during program execution. A variable is only a name given to a memory location, all the operations done on the variable effects that memory location. In C++, all the variables must be declared before use.

To know more about C++ variables, refer to this article – C++ Variables

6. Variable Scope in C++

In general, the scope is defined as the extent to which something can be worked with. In programming also the scope of a variable is defined as the extent of the program code within which the variable can be accessed or declared or worked with. There are mainly two types of variable scopes, Local and Global Variables.

To know more about variable scope in C++, refer to this article – Scope of Variable in C++

7. Uninitialized Variable in C++

"One of the things that have kept C++ viable is the zero-overhead rule: What you don't use, you don't pay for." -Stroustrup. The overhead of initializing a stack variable is costly as it hampers the speed of execution, therefore these variables can contain indeterminate values. It is considered a best practice to initialize a primitive data type variable before using it in code.

To know more about what happens to the uninitialized variables, refer to this article – Uninitialized primitive data types in C/C++

8. Constants and Literals in C++

As the name suggests the name constants are given to such variables or values in C++ programming language which cannot be modified once they are defined. They are fixed values in a program. There can be any type of constant like integer, float, octal, hexadecimal, character constants, etc. Every constant has some range. The integers that are too big to fit into an int will be taken as long. Now there are various ranges that differ from unsigned to signed bits. Under the signed bit, the range of an int varies from -128 to +127, and under the unsigned bit, the int varies from 0 to 255. Literals are a kind of constant and both terms are used interchangeably in C++.

To know more about constants in C++, refer to this article – Constants in C++

To know more about literals in C++, refer to this article – Literals in C++

9. Operators in C++

Operators are the foundation of any programming language. Thus the functionality of the C/C++ programming language is incomplete without the use of operators. We can define operators as symbols that help us to perform specific mathematical and logical computations on operands. In other words, we can say that an operator operates the operands.

To know more about C++ operators, refer to this article – Operators in C++

10. Loops in C++

Loops in programming come into use when we need to repeatedly execute a block of statements. For example: Suppose we want to print "Hello World" 10 times. This can be done in two ways, the Iterative method and using Loops.

To know more about loops in C++, refer to this article – C++ Loops

11. Decision-Making in C++

There comes situations in real life when we need to make some decisions and based on these decisions, we decide what should we do next. Similar situations arise in programming also where we need to make some decisions and based on these decisions we will execute the next block of code. Decision-making statements in programming languages decide the direction of the flow of program execution.

To know more about decision-making statements in C++, refer to this article – Decision Making in C++

12. Classes and Objects in C++

Classes and Objects are used to provide Object Oriented Programming in C++. A class is a user-defined datatype that is a blueprint of an object that contains data members (attribute) and member methods that works on that data. Objects are the instances of a class that are required to use class methods and data as we cannot use them directly.

To know more about classes and objects, refer to this article – C++ Classes and Objects

13. Access Modifiers in C++

Access modifiers are used to implement an important feature of Object-Oriented Programming known as Data Hiding. Access modifiers or Access Specifiers in a class are

used to set the accessibility of the class members. That is, it sets some restrictions on the class members not to get directly accessed by outside functions.

To know more about access modifiers in C++, refer to this article – Access Modifiers in C++

14. Storage Classes in C++

Storage Classes are used to describe the features of a variable/function. These features basically include the scope, visibility, and lifetime which help us to trace the existence of a particular variable during the runtime of a program.

To know more about storage classes in C++, refer to this article – Storage Classes in C++

15. Forward declarations in C++

It refers to the beforehand declaration of the syntax or signature of an identifier, variable, function, class, etc. prior to its usage (done later in the program). In C++, Forward declarations are usually used for Classes. In this, the class is pre-defined before its use so that it can be called and used by other classes that are defined before this.

To know more about forward declarations in C++, refer to this article – What are forward declarations in C++?

16. Errors in C++

An error is an illegal operation performed by the user which results in the abnormal working of the program. Programming errors often remain undetected until the program is compiled or executed. Some of the errors inhibit the program from getting compiled or executed. Thus errors should be removed before compiling and executing.

To know more about C++ errors, refer to this article – Errors in C++

17. Undefined Behaviour in C++

If a user starts learning in a C/C++ environment and is unclear about the concept of undefined behavior then that can bring plenty of problems in the future while debugging someone else code might be actually difficult in tracing the root to the undefined error.

To know more about undefined behavior, refer to this article – Undefined Behavior in C and C++

549