The Direct Path to

C++

Programming Language

Hamidullah Lutfy
Wahidullah Lutfy

Table of Contents

C++ Objects (instances of classes)
Using member functions of C++ objects

Chapter 10

File System in general
Input and Output I/O with C++ fstream module/class
The ifstream module/class for Input Files
The ofstream module/class for Output Files

Chapter 11

Introduction to Data Structures
Arrays in C++
The string module/class
Vector in C++
Class in C++
Struct and Link List in C++
Map or Data Dictionary in C++

Chapter 12

Comparison of C++ Data Structures with Python

Chapter 13

Pointers in C++
References in C++
What is the Difference between pointers and reference

Chapter 14

C++ Compilers
C++ Integrated Development Environment (IDE)
Compilation of C++ in different platforms

Chapter 15

Debugging C++ source code using mdb
Automatic compilation in C++ using make utility

Chapter 16

Introduction to Web Programming using C++
Using C++ to run any operating System Commands

Chapter 17

The C++ Object Oriented (OO) Paradigm

- Abstraction (Data Hiding)
- Encapsulation
- Inheritance

- Polymorphisim

Chapter 18

Exception Handling
- How to handle exceptoin
- Ways and when to Throw an Exception
- Multiple Throws and Catches
- The try-catch block
- Assertion and Debugging your program

Chapter 19

The C++ Template or Class
- Template to for overloading functions
- Template for data hiding and abstraction

Chapter 20

The C++ references
Finding C++ Documentation using IDE
Finding C++ documentation using the internet
Finding C++ documentation using UNIX, Linux, Mac

Chapter 21

The future of C++ Programming Language
C++ still evolving
Further Research in C++ Programming

APPENDICES

The ASCII Character Set
The Unicode Character Set
The C++ Reserved keywords
The C++ Precedence of Operators
Some of the standard C++ Modules (Classes)
Some of the standard or predefined C++ Functions
The C++ Function Overloading
The C++ Operator Overloading
The Inline Function
The Friend Function
The Static Function
The C++ Recursion Function

Preface

About this book

In this class "**The Direct Path to C++ Programming Language**", I would like to provide you with a clear path to learning the "C++ Programming Language" My Objective is to first introduce you to the "**syntax** and **semantics**" of the **"C++ Programming" ,** as well as provide you with a precise definition of the topic I am covering in each section. Then, by providing further examples and explanation I make sure you have a clear path to understanding the C++ programming language. You will also be given lab activities that you could do it on your computer with your favorite operating system and C++ compiler of choice or C++ Integrated Development Environment (IDE) that you prefer to use.

While it is physically not possible to mentor you with the lab activities while you are using my book, it is absolutely essential for you to be your own mentor whenever you do the lab activities or learning anything in life. Generally, there is no better mentor or coach than ourselves! The first path to successful learning in C++ starts by memorizing what we read in each chapter, practice them by writing the snippet of the source code, compile and run it. If we see the sample dialogue in the output of our program that matches the output of that particular example from the book, then we are learning successfully and should continue memorizing what we did and test it using a different operating system and a different C++ compiler on a different day, if we still are successful at it, congratulation you are doing great!

However, if you find out the sample dialogue output of your program is different than the textbook, maybe you inadvertently had a typo that is having a syntax error during the compilation, or maybe you are not getting the same output due to your program has a logical error or semantic error, which your code meaning is different than what you intended to do. For instance you wrote if (x = y), when you meant to use the comparison "==" operator and instead you used the assignment "=" operator. The compiler will not report that as an compilation error since the syntax is correct, but at run time it will have a different meaning when it runs dynamically. By the way, I just explained the difference between "**Syntax errors** or static errors or compile time errors" vs the "S**emantic errors** or dynamic error or run time errors". More about that will be explained in subsequent chapters.

Chapter 1

Introduction to Computer Programming

Why we program?

What is a workflow?

What is an algorithm?

What are the different types of computer programming languages?

What is a source code?

What is a compiler and what is an IDE?

What is an Operating System?

What are examples of Software**?**

Chapter 2

Setup your C++ Development Environment
How to setup your sandbox or Windows platform
How to setup your sandbox for UNIX and Linux platform

Chapter 3

Introduction to C++ Programming

Your first “Hello, World!” Program in C++
Writing your source-code using the C++ Compiler Collection
Naming your first program “helloWorld.cpp”
Compiling and running your first program using the Gnu C++ “g++” compiler
Running your first program
The details of the “helloWorld.cpp” is explained line by line

Your first “Hello, World!” Program in C++
Here is the C++ source code for your first “helloWorld.cpp” program

Example 1:
// This is a single line comment in C++

```
// This is  my first C++ program
/*
I can also write multi-line comment  using the /* … */  to document my first program
Program Name: helloWorld.cpp
Program Description: This is my first program that will display the string “Hello, World!” to
the terminal console as the output.
Written by: Wahidullah Lutfy
*/
#include <iostream>

using namespace std;

int main()
{

    cout << “Hello, World!” << endl;
    return (0);
}
```

Here is the same program, this time we are commenting the **“using namespace std”** and use the scope resolution “**std::** ” to resolve naming conflicts for objects such as console output **“cout”** and **“endl”** in this first simple program.

Example 2:

```
// This is a single line comment in C++
// This is  my first C++ program
/*
I can also write multi-line comment  using the /* … */  to document my first program
Program Name: helloWorld.cpp
Program Description: This is my first program that will display the string “Hello, World!” to
the terminal console as the output.
Written by: Wahidullah Lutfy
*/
#include <iostream>

/* By putting the ‘two slashes ‘//’ in front of any line, C++ compiler will ignore it
and thread that line as comment.
In the example we could simply delete that line or comment it like we did here.
*/
//using namespace std;

int main()
{
    std::cout << “Hello, World!” << std::endl;
```

```
    return (0);
}
```

Now, in our third example we remove all the comments (both **single line comment '//'** and **multi-line /* … */ comment** including the line that we commented in the example 2 that was for **"// using namespace std:"**.

Example 3:

```
#include <iostream>

int main()
{
    std::cout << "Hello, World!" << std::endl;
    return (0);
}
```

As you can see from the above example 3, we have removed all the comments. We use comments to make our program more readable by adding the necessary comments and proper indentation our program is much more readable and understandable to ourselves when we look at it at a later date and are more appreciated by other developers who may work with us on the same project or development team.

Lab Activity:

Chapter 4

C++ Basics

Variables and Statements

Chapter 5

C++ Program Layout

Compiling C++ Programs

Chapter 6

C++ PreProcessors Directives

The C++ I/O iostream module (class)

The Scope Resolution "using namespace std"

Chapter 7

Flow of Control in C++

Branching in C++ (if statement and switch statement)

Looping in C++ (for, while, and do while statements)

Chapter 8

Functional Programming in C++

Object Oriented Programming C++

Chapter 17

The C++ Object Oriented (OO) Paradigm

- Abstraction (Data Hiding)
- Encapsulation
- Inheritance
- Polymorphisim

Chapter 18

Exception Handling

How to handle exceptoin
Ways and when to Throw an Exception
Multiple Throws and Catches
The try-catch block
Assertion and Debugging your program

Chapter 19

The C++ Template or Class

Template to for overloading functions
Template for data hiding and abstraction

Chapter 20

The C++ references
Finding C++ Documentation using IDE
Finding C++ documentation using the internet
Finding C++ documentation using UNIX, Linux, Mac

Chapter 21

The future of C++ Programming Language
C++ still evolving
Further Research in C++ Programming

APPENDICES

The ASCII Character Set
The Unicode Character Set
The C++ Reserved keywords
The C++ Precedence of Operators
Some of the standard C++ Modules (Classes)
Some of the standard or predefined C++ Functions
The C++ Function Overloading
The C++ Operator Overloading
The Inline Function

The Friend Function
The Static Function
The C++ Recursion Function