Complete C++ Series:

A Quick overview of how I have structured this course:

- ❖ Basics :
  - ➢ Fundamentals of programming in C++
  - ➢ Data Types
  - ➢ Decision Making
  - ➢ Loops
  - ➢ Functions
- ❖ Intermediate:
  - ➢ Arrays
  - ➢ Pointers
  - ➢ Structures
  - ➢ Enumerations
  - ➢ Streams
- ❖ Advanced:
  - ➢ Classes
  - ➢ Exceptions
  - ➢ Templates
  - ➢ Containers

How have I structured this Course ?

I have structured this course for absolute beginners who are willing to develop real-world applications with C++ e.g Desktop. So by the end of the course, you will have a solid understanding of C++ and you will be ready to apply it in real life for example: you want to build games with **Unreal Engine**, which is a popular gaming engine, you will have the necessary knowledge about C++ and you will only have to learn about **Unreal Engine** if that's what you want 🤭.

Introduction tO C++

Before we start coding, let's spend a couple of minutes and let's gossip about C++, What you can do with it ? and How to master it 💀 ?

C++ is one of the popular programming languages in the world and it's the language of choice for building:

- High-performance applications
- Video Games
- Device Drivers
- Web Browsers
- Servers
- Operating Systems etc...

That's why is used by large companies like:

- Abode
- Google
- Microsoft
- Netflix
- And even government companies like NASA

Benefits of Learning C++:

- **Build Cool Stuff**: Learning C++ helps you create really cool things like video games, robots, and even apps for your computer. It's like having a superpower to make whatever you can imagine!
- **Understand Computers Better**: C++ teaches you how computers work inside. It's like peeking under the hood of a car to see how it runs. You'll learn about memory, how programs talk to hardware, and other cool tech stuff.
- **Problem-Solving Skills**: C++ teaches you how to solve problems step by step, just like solving puzzles. This helps you become smarter and better at figuring out tricky challenges, like how to make a game work perfectly or fix a computer bug.
- **Join a Big Community**: Lots of people around the world love C++, so when you learn it, you join a big community of friends who can help you and share cool ideas. It's like being part of a big team where everyone is super smart and loves technology!
- **Unlock Job Opportunities**: When you're older, knowing C++ can help you get really cool jobs where you make lots of money doing what you love. Companies are always looking for smart people who know C++ to help them build amazing things, so it's like having a golden ticket to a bright future!

Learning C++ is a great investment and opens a lot of doors for you. According to indeed.com the average salary of an entry level C++ developer annually is $117,695.

Now to Master 💀 C++ there are two things you need to learn:

1. C++ Language : meaning the syntax or grammar of the language

```cpp
#include <iostream>
using namespace std;


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

2. C++ Standard Library or STL which is a collection of pre-written C++ code that provides functionalities that is required by many applications examples are: Data Structures (list, Map) and Algorithms(for Searching and Sorting Data) don't worry about them now. We will talk about them later 🤭. However this functions are required in almost every application so instead of us creating all this functionality from scratch every single time we can reuse some of this C++ code in those libraries to quickly build applications 😎.

Myth about C++

A Lot of people find C++ a bit extensive and intimidating but in reality you don't need to learn all about C++ to be able to write substantial programs for the same reason, you don't need to know all of the features your TV provides just to use and enjoy it.

SO C++ IS NOT DIFFICULT OK 🐱🏍️.

~~~~~~~~~~~~~

So in this course, we're going to explore C++ step-by-step and as we go, am going to show how to write some really cool programs as you are learning C++ plus am going to give you plenty of exercises to help you better understand and remember the concepts. Then you see that C++ is not really that difficult. So if you follow along you, by the end of this course you will be able to write C++ programs with Confident 👨🏼💪🏽.
~~~~~~~~~~~~~

What do you need to write C++ programs ?

Just like when you want to type documents you will have to use applications like notepad, ms word etc.. in the case of C++ you will need an application called IDE short for INTEGRATED DEVELOPMENT ENVIRONMENT. Which is basically an editor for writing code as well as build and debugging tools. there are lots of IDEs out there both commercial and free once examples are:

- Ms Visual Studio
- XCODE (App Store for Mac Users)
- CLION
- Dev C++ (download version 5.11)

You can use any of these free alternatives, depending on the Operating System you are using. But here in this course, We will be using Dev C++ for writing and compiling our code ok.

Your First C++ Program

Let's start with a metaphor, think of your TV, your TV has several functions, it has functions for changing the channel, controlling the volume and so on. By the same token a C++ program consists of tens, hundreds or even thousands of functions each serving a purpose.

```
main()
```

Now we have a special function called "**main()**" which is the entry point to our program. Which is like the power button of the TV. Note that C++ is a case sensitive language so it's sensitive to upper and lowercase letters. So make sure to type the "**main()**" function exactly as I show you. Else if you type **"Main()"** it's going to mean something completely different to C++ ok.

```
int main()
```

When declaring a function we have to specify the type of value is going to return. Our "**main()**" function should return a value of type "**int**"

which is short for integer (whole number e.g 0, 1, 2, 3 and so on). So when we run our program the operating system like windows, macOS or linux is going to execute this function, and the value that this function returns tells the operating system whether the program compiled successfully or not.

```
int main() {}
```

After the function declaration we type parenthesis **"()"** inside of those parenthesis, we can type parameters because a function can have zero or more parameters but for now let's not worry about them. Now after we close the parenthesis, we type curly braces. Inside of those braces is where we type what we want the function to do. So whatever we type in between the brace **"{}"** will tell the operating system what should happen when we run the program ok. In this function we want to write code to print something on the screen when we run our program for that we use gonna use the **C++ STL (short for C++ Standard library).** So earlier I told you told that the standard library contains a bunch of capabilities we need in almost every application.

```
#include <iostream>


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

On the top we have **"#include <iostream>"**. **"Iostream"** is short for **Input Output Stream**. Iostream is the actual name of the file but we specify it in angle brackets to tell the computer that it's a standard library. In that file we have capabilities for printing something to the screen or getting input from the user. Just like a supermarket has different sessions, the standard library also has different files each serving a purpose.

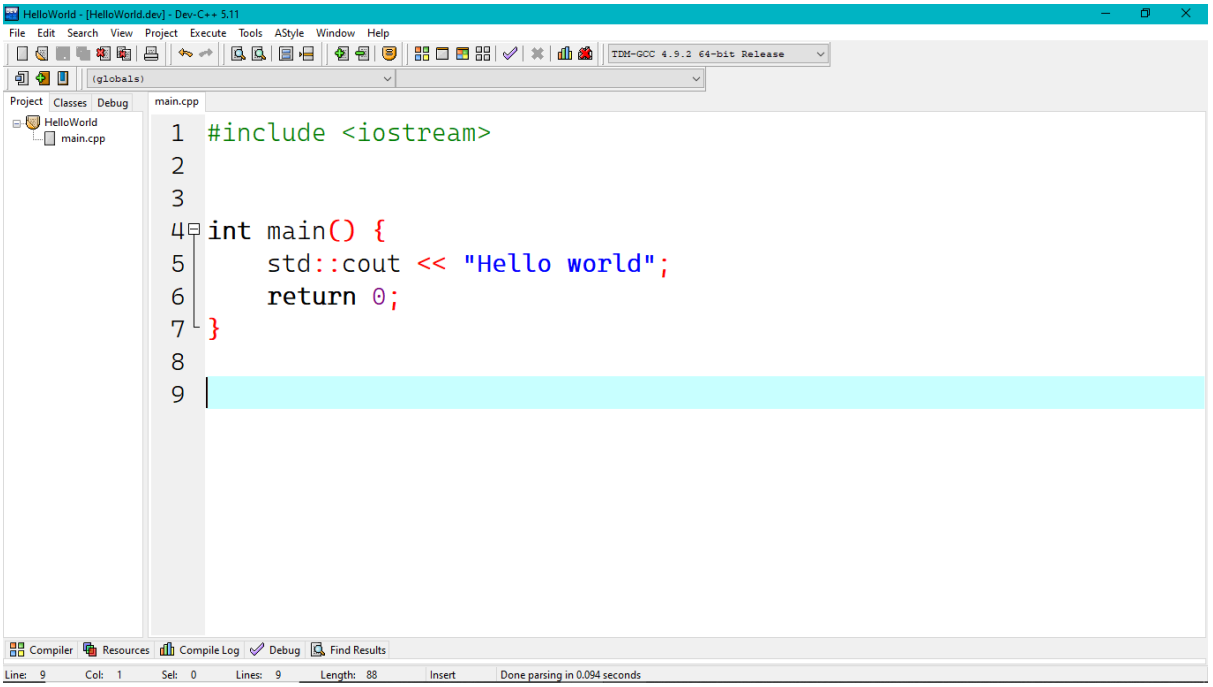
```
std::cout << "Hello world";
return 0;
```

The **std** namespace is like a bucket or features that are currently available to us. To enable us use those features embedded in the **std** namespace we type double Caret **"::"** after the std. There are tons of features available however, in this course we will be using **"cout"** which is short for **character out.** Using that feature/object we can output one or more characters to the screen after a double less than sign **"<<"** However this is called **stream inspection** operator in C++. Then we terminate the it with a semicolon **";"**. In C++, we refer to that as a **statement**. A **statement** in C++ is a piece of code that produces a value. So the value that our program outputs/produces is **"Hello world"**. Finally we return 0, but why 0 ? Well zero tells our Operating system that our program is going to terminate correctly, if we pass any other value positive or negative it means our program encountered an error ok.

```
#include <iostream>


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

Congratulations, this is ur first C++ Program. Now let's compile and run the program with Dev C++ IDE.

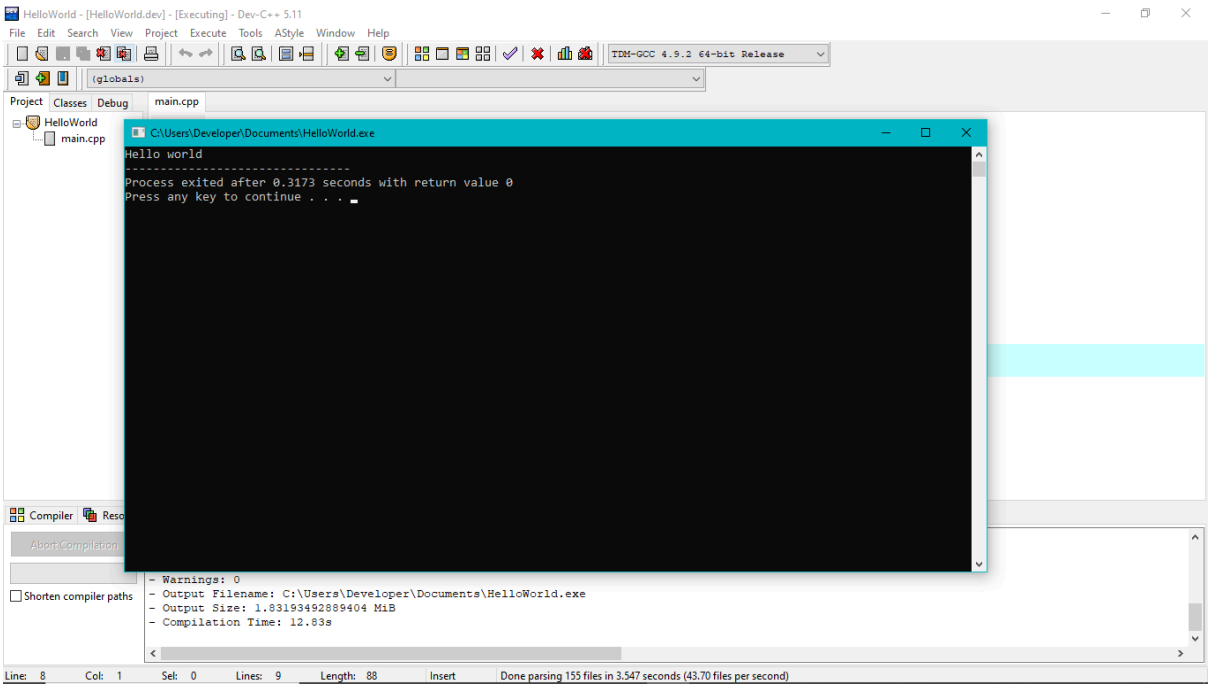


In this IDE we will be using the File, Edit, Project and Execute menu more, so don't worry about those little icons ok. In the Execute menu you will find :

- Compile:
  - Description: It's used to compile your C++ code into an executable program.
  - Shortcut Key: F9
- Run:
  - Description: Under the Execute menu, you'll see the "Run" option. This is used to execute/run the compiled program and view the output.
  - Shortcut Key: Ctrl + F10
- Compile and Run:
  - Description: In the Execute menu, you'll find the "Compile & Run" option. This compiles the code and immediately runs the compiled executable.
  - Shortcut Key: F10
- Rebuild:
  - Description: Within the Execute menu, there's the "Rebuild All" option. It cleans existing compiled files and then compiles all files again from scratch.
  - Shortcut Key: Ctrl + F11

Compiling and Running a Program

To run our program, first we need to translate the program into machine code (0, 1). This machine code is native language that the computer understands and it's different from one operating systems to the other. So if we compile our code on windows, we get an executable but that executable will only run on windows, so we cannot run that on macOS or linux. However if we want to get an executable for mac or linux, we will have to take that code and compile it to get an executable for mac or linux ok. Let's run our program on windows :



Here I've Compiled & Run my program with shortcut keys as listed above. Always use shortcut keys because it makes your life easier. The blackscreen you see shows that it's a console application or terminal and it's basically a way of showing the output of our program. If your code doesn't get compiled successfully it's completely normal, just ensure to follow along with whatever I've typed and I promise if you pay close attention, you will be able to solve issues on your own ok 💪🏽.