Basics of C++

By: Aaryan Vinod Kumar

Why Learn C++?

C++ is a **powerful, high-performance programming language** that combines the features of both low-level and high-level languages. It's widely used in **competitive programming**, **system-level development**, and especially **DSA**, due to its control over memory and fast execution.

C++ gives you:

- **Speed** (close to C)
- **Object-oriented programming**
- Rich **Standard Template Library (STL)** for built-in data structures

Learning this programming language is **not hard.** At first, people may feel that it makes no sense at all. Whenever that happens, stop trying to read it like english language, and see it as **a set of steps**.

How Does C++ Work?

1. **Write Code** → You write human-readable `.cpp` code.

2. **Compile** → A compiler (like g++) translates it into machine code.

3. **Execute** → The machine code is run by the computer.

C++ is a **compiled language***, which means it checks errors before running and runs faster than interpreted ones like Python.

Compiled language: Reads the whole code at once and presents all the errors in one go, rather than an interpreting language which reads the code line-by-line

In C++, you find something knows as a Standard Template Library (STL). It is a powerful set of **predefined classes and functions** in C++ that helps you work with common **data structures and algorithms** without writing everything from scratch.

Think of it as your **DSA toolbox**; Everything you would ever require is there, readily accessible.

Why STL is Awesome for DSA:

- Saves **time and effort**
- Ensures **optimized performance**
- Lets you focus on **logic**, not low-level implementation
- Helps in **competitive programming** and **interviews**

Variables & Data Types:

C++ requires you to declare the type of data a variable will store. Here are the most common ones:

Data Type	Use	Example
`int`	Whole numbers	`int age = 20;`
`float`	Decimal (less precise)	`float pi = 3.14;`
`double`	Decimal (more precision)	`double gravity = 9.8;`
`char`	Single character	`char grade = 'A';`
`string`	Text (needs `<string>` header)	`string name = "Aaryan";`
`bool`	Boolean true/false	`bool passed = true;`
`long`	Larger integers	`long population = 1000000;`
`long long`	Very large numbers	`long long dist = 1e18;`
`unsigned`	Only non-negative values	`unsigned int score = 42;`

Programs in C++:

Now, lets see how most programs in C++ look like:

Program to print “Hello World!”

```cpp
#include <iostream>

using namespace std;



int main() {

    cout << "Hello world!";

    return 0;

}
```

Lets see what each line means:

1. `#include <iostream>`

   This statement is basically like taking a tool out of the Standard Template Library toolbox. In this case, iostream is the tool that we are taking out, which allows us to take input from the user and give an output.

2. `Using namespace std;`

   In C++, all standard library features like `cout`, `cin`, `string`, etc., belong to the **std** namespace. So, without `using namespace std;`, you'd have to write:

   ```cpp
   std::cout << "Hello";

   std::cin >> name;
   ```

By writing `using namespace std;`, you're telling the compiler, **"Hey, I'll be using things from the `std` namespace a lot, so don't make me write `std::` every time."**

It's just a shortcut to make your code cleaner and faster to write, especially for beginners.

3.
```
int main () {
    cout << "Hello, world!";
    return 0;

}
```

This is the main part of the program. Basically, **int main()** is a function inside of which you write your whole code logic, what it does, what it takes in, what it prints out etc. This function is always executed at runtime, i.e, whenever you run the program (hence its called the main).

cout: Read this as see-out, this is known as **character output**. This is basically instructing the computer to output whatever follows the "`<<`".

You can even think of "`<<`" as saying "Send this into cout."

return 0: This is just to tell the Operating System that the program has executed successfully. It is not necessary in a code, but a good practice.

Why 0 though?

If you noticed at the start, we have written **int main()**, and what that tells to the OS is that whatever comes to me will be an integer value. This will come in detail later, when I pick up functions in C++. For now, just remember that wherever you see **int,** it means that it only deals with integral values.

Program to greet the user

```cpp
#include <iostream>

using namespace std;



int main() {

    string name;

    cout << "Enter your name: ";

    cin >> name;

    cout << "Hello, " << name << "!" << endl;

    return 0;

}
```

Let's break this down:

1. `string name;`

   Here, we are declaring a variable of type **string** called **name**. This will store the value that the user enters. The **string** data type is used for storing text (words, sentences, etc.). It's important to remember that C++ is strongly typed, meaning we have to specify the type of data a variable will store.

2. `cout << "Enter your name: ";`

   We use **cout** to output a message asking the user to enter their name. The **<<** is pushing this text into the **cout** stream to display on the screen.

3. `cin >> name;`

   Here is where we take input from the user. **cin** is used to read input from the user. The **>>** operator is used to extract the input and store it in the variable **name**.

   1. You can think of **cin** as "see-in," like you're telling the program to look for data from the user.
   2. The **>>** operator takes the input provided by the user and stores it in the variable **name**.

After taking the input, we use **cout** again to print out the greeting, "Hello, **<name>**". We combine the string **"Hello, "** with the value stored in the **name** variable using **<<** to create the output.

Just like before, **return 0;** indicates that the program has completed successfully.

Well, if you made through all that, woohoo!!

Try these programs in your own compiler, make some random changes, see where they take you.