LAB # 1

Introduction to C++

Introduction to C++

C++, as we all know is an extension to C language and was developed by **Bjarne stroustrup** at bell labs. C++ is an intermediate level language, as it comprises a confirmation of both high level and low level language features. C++ is a statically typed, free form, multiparadigm, compiled general-purpose language.

C++ is an **Object Oriented Programming language** but is not purely Object Oriented. Its features like Friend and Virtual, violate some of the very important OOPS features, rendering this language unworthy of being called completely Object Oriented. Its a middle level language.

Benefits of C++ over C Language

The major difference being OOPS concept, C++ is an object oriented language whereas C language is a procedural language. Apart form this there are many other features of C++ which gives this language an upper hand on C laguage.

Following features of C++ makes it a stronger language than C,

1. There is Stronger Type Checking in C++.
2. All the OOPS features in C++ like Abstraction, Encapsulation, Inheritance etc makes it more worthy and useful for programmers.
3. C++ supports and allows user defined operators (i.e Operator Overloading) and function overloading is also supported in it.
4. Exception Handling is there in C++.
5. The Concept of Virtual functions and also Constructors and Destructors for Objects.
6. Inline Functions in C++ instead of Macros in C language. Inline functions make complete function body act like Macro, safely.
7. Variables can be declared anywhere in the program in C++, but must be declared before they are used.

How to open C++ in Microsoft Visual Studio :

**Microsoft Visual C++** (often abbreviated to **MSVC**) is an [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) product from [Microsoft](https://en.wikipedia.org/wiki/Microsoft) for the [C](https://en.wikipedia.org/wiki/C_(programming_language)), [C++](https://en.wikipedia.org/wiki/C%2B%2B), and [C++/CLI](https://en.wikipedia.org/wiki/C%2B%2B/CLI) [programming languages](https://en.wikipedia.org/wiki/Programming_language). MSVC is [proprietary software](https://en.wikipedia.org/wiki/Proprietary_software); it was originally a standalone product but later became a part of [Visual Studio](https://en.wikipedia.org/wiki/Microsoft_Visual_Studio) and made available in both [trialware](https://en.wikipedia.org/wiki/Trialware" \o "Trialware) and [freeware](https://en.wikipedia.org/wiki/Freeware) forms. It features tools for [developing](https://en.wikipedia.org/wiki/Software_development) and [debugging](https://en.wikipedia.org/wiki/Debugging) C++ code, especially code written for the [Windows API](https://en.wikipedia.org/wiki/Windows_API), [DirectX](https://en.wikipedia.org/wiki/DirectX) and [.NET](https://en.wikipedia.org/wiki/.NET_Framework).

Many [applications](https://en.wikipedia.org/wiki/Application_software) require [redistributable](https://en.wikipedia.org/wiki/Redistributable) Visual C++ packages to function correctly. These packages are often installed independently of applications, allowing multiple applications to make use of the package while only having to install it once. These Visual C++ redistributable and runtime packages are mostly installed for standard [libraries](https://en.wikipedia.org/wiki/Library_(computing)) that many applications use.

### To create a project and add a source file

1. Create a project by pointing to **New** on the **File** menu, and then clicking **Project**.
2. In the **Visual C++** project types pane, click **Win32**, and then click **Win32 Console Application**.
3. Type a name for the project.

By default, the solution that contains the project has the same name as the project, but you can type a different name. You can also type a different location for the project.

Click **OK** to create the project.

1. In the **Win32 Application Wizard**, click **Next**, select **Empty Project**,and then click **Finish**.
2. If **Solution Explorer** is not displayed, on the **View** menu, click **Solution Explorer**.
3. Add a new source file to the project, as follows.
   1. In **Solution Explorer**, right-click the **Source Files** folder, point to **Add**, and then click **New Item**.
   2. In the **Code** node, click **C++ File (.cpp)**, type a name for the file, and then click **Add**.

The .cpp file appears in the Source Files folder in **Solution Explorer**, and the file is opened in the Visual Studio editor.

1. In the file in the editor, type a valid C++ program that uses the Standard C++ Library, or copy one of the sample programs and paste it in the file.

For example, you can use the [set::find](https://msdn.microsoft.com/en-us/library/ets82w6a.aspx) sample program, which is one of the the Standard Template Library samples that are included in Help.

If you use the sample program, notice the using namespace std; directive. This directive enables the program to use cout and endl without requiring fully qualified names (std::cout and std::endl).

1. Save the file.
2. On the **Build** menu, click **Build Solution**.

The **Output** window displays information about the compilation progress, for example, the location of the build log and a message that indicates the build status.

1. On the **Debug** menu, click **Start without Debugging**.

If you used the sample program, a command window is displayed and shows whether certain integers are found in the set.

## Your first C++ program

## Now you have installed the compiler based on your OS, it’s time to write your first C++ program.

### “Hello World!”

Your first C++ program will be a “Hello World!” program.

You might have noticed “Hello World!” being the first program while starting out with any programming language. This is because:

* It is a standard check to see whether everything is working fine or not.
* There will be very less code to start with.
* The less code makes it intuitive for the beginners to get acquainted with the language.
* The code is enough to learn the basic syntax and semantics of the language.

So, let’s get coding.

#include <iostream>

using namespace std;

int main()

{

cout<<"Hello World!";

return 0;

}

The program prints Hello World! in the output screen.

### How the program works?

Now, let’s dissect the above code. The code is divided into six major parts:

* #include <iostream>
* using namespace std
* ;
* int main() { }
* cout << “Hello World!”;
* return 0;

1. **What is #include <iostream>?**  
     
   If you’ve already written code in C language before, you might seen this line of code before. If you haven’t, don’t worry we’ll cover it now.  
     
   This statement includes the header file into the application so that you are able to use the operations included in them. Also, you can create your own header files and include them in your program using the #include.  
     
   **What is iostream?**

iostream is what you call the header file. It is a standard C++ input/output library file.  
It comes packaged with the compiler/IDE and contain mechanisms to get the information from the user and print same or added information to a file, screen or any other media.

**What is #include?**

The #include iostream file, into the program. This ensures that now you’re able to use the operations, iostream operations (like: taking input from user, displaying output on the screen), in the program.

1. **What is using namespace std;”?**  
     
   The statement is intuitive in itself, you are “using” the “namespace” “std” in your file.  
   We use the namespace std to make it easier to reference operations included in that namespace.  
   If we hadn’t used the namespace, we’d have written **std::cout** instead of **cout**. This tells the compiler that every **cout** is actually **std::cout**.  
     
   **What’s a namespace?**  
     
   It’s a region where your code resides. It limits or expands the scope of your code to one or more files.  
     
   **Why do you use namespace?**  
     
   Like two persons can have the same name, variables and functions in C++ can have same names as well. The use of namespace is to avoid the confusion of which variables/functions you are referencing to.  
     
   **What is std?**  
     
   std is a standard namespace used in C++.
2. **Semicolon ”;”**  
     
   Ask any C++ programmer and they will tell you at least one horror story related to the semicolon ; .  
     
   The semicolon is a terminal. It terminates a statement. When missed or incorrectly used, it will cause a lot of issues.
3. **int main() { }**  
     
   As the name suggests, it is the main function of the program. The code inside { } is called the body and is executed first when you run your C++ program.  
     
   It is one code that is mandatory in a C++ program. If you just have this line of code alone, your program will be valid.
4. **cout << “Hello World!”;**  
     
   This statement prints “Hello World!” onto the output screen.  
     
   The cout is an object of standard output stream. What this means is, it outputs/prints the data after *<<* , i.e. Hello World! into a stream (in this case, the output screen).  
     
   **What is a stream?**  
     
   Stream is basically a sequence of objects, usually bytes. It can describe files, input/output terminal, sockets, etc.  
   **What is <<?**  
     
   << is the insertion operator used to write formatted data into the stream.
5. **What is return 0;?**  
     
   This statement returns 0 ‘zero’.  
     
   This is called a return statement. It isn’t mandatory to return anything from the main() function but is rather a convention. If not return, the compiler returns a status automatically.  
     
   **Why zero in return statement?**  
     
   It denotes Exit status of the application that basically the tells system “The program worked fine.”