Lab-1 Switching from Java to C++

C++ is a programming language created by Bjarne Stroustrup and his team at Bell Laboratories in 1979. Forty years later, it is one of the most widely used languages. **C++** runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX.

Advantages of C/C++

There are 1000s of good reasons to learn C++ Programming. **C++** is a MUST for students and working professionals to become a great Software Engineer. Some of the key advantages of learning C++:

- C++ is a statically typed, compiled, general-purpose, case-sensitive, free-form programming language that supports procedural, object-oriented, and generic programming.
- C++ is regarded as a **middle-level** language, as it comprises a combination of both high-level and low-level language features.
- C++ is very close to hardware, so you get a chance to work at a low level which gives you lot of control in terms of memory management, better performance and finally a robust software development.
- C++ is one of the ever-green programming languages and loved by millions of software developers.
- C++ is the most widely used programming languages in application and system programming.
- C++ teaches you the difference between compiler, linker and loader, different data types, storage classes, variable types their scopes etc.

Standard Libraries

Standard C++ consists of three important parts −

- The core language giving all the building blocks including variables, data types and literals, etc.
- The C++ Standard Library giving a rich set of functions manipulating files, strings, etc.
- The Standard Template Library (STL) giving a rich set of methods manipulating data structures, etc.

Installing GNU C/C++ Compiler

o install GCC at Windows you need to install MinGW. To install MinGW, go to the MinGW homepage, www.mingw.org, and follow the link to the MinGW download page. Download the latest version of the MinGW installation program which should be named MinGW-<version>.exe.

While installing MinGW, at a minimum, you must install gcc-core, gcc-g++, binutils, and the MinGW runtime, but you may wish to install more.

Add the bin subdirectory of your MinGW installation to your **PATH** environment variable so that you can specify these tools on the command line by their simple names.

When the installation is complete, you will be able to run gcc, g++, ar, ranlib, dlltool, and several other GNU tools from the Windows command line.

Hello World Program in C++

```
// Single line Comment

/*
* Multiple line
* comment
*/
  1. #include<iostream>
  2. using namespace std;
  3. int main()
  4. {
  5. cout<<"Hello World!";
  6. return 0;
  7. }
```

The output will be **"Hello World!"**.

Let's discuss each and every part of the above program.

Line 1:

#include<iostream>

This statements tells the compiler to include iostream file. This file contains pre defined input/output functions that we can use in our program.

Line 2:

using namespace std;

A namespace is like a region, where we have functions, variables etc and their scope is limited to that particular region. Here std is a namespace name, this tells the compiler to look into that particular region for all the variables, functions, etc. I will not discuss this in detail here as it may confuse you. I have covered this topic in a separate tutorial with examples. Just follow the tutorial in the given sequence and you would be fine.

Line 3:

int main()

As the name suggests this is the main function of our program and the execution of program begins with this function, the int here is the return type which indicates to the compiler that this function will return a integer value. That is the main reason we have a return 0 statement at the end of main function.

Line 5:

cout << “Hello World!”;

The `cout` object belongs to the iostream file and the purpose of this object is to display the content between double quotes as it is on the screen. This object can also display the value of variables on screen(don’t worry, we will see that in the coming tutorials).

Line 6:

return 0;

This statement returns value 0 from the main() function which indicates that the execution of main function is successful. The value 1 represents failed execution.

C++ Variables

Variables are containers for storing data values.

In C++, there are different **types** of variables (defined with different keywords), for example:

- int - stores integers (whole numbers), without decimals, such as 123 or -123
- double - stores floating point numbers, with decimals, such as 19.99 or -19.99
- char - stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes

- string - stores text, such as "Hello World". String values are surrounded by double quotes
- bool - stores values with two states: true or false

Example

```
int myNum = 5;                // Integer (whole number without decimals)
double myFloatNum = 5.99;     // Floating point number (with decimals)
char myLetter = 'D';          // Character
string myText = "Hello";      // String (text)
bool myBoolean = true;        // Boolean (true or false)
```

Display Variables

The cout object is used together with the << operator to display variables.

To combine both text and a variable, separate them with the << operator:

Example

```
int myAge = 35;
cout << "I am " << myAge << " years old.";
```

Constants

```
const int myNum = 15;  // myNum will always be 15
myNum = 10;  // error: assignment of read-only variable 'myNum'
```

String Types

The string type is used to store a sequence of characters (text). This is not a built-in type, but it behaves like one in its most basic usage. String values must be surrounded by double quotes:

Example

```
string greeting = "Hello";
cout << greeting;
```

To use strings, you must include an additional header file in the source code, the <string> library:

Example

```
// Include the string library
#include <string>

// Create a string variable
string greeting = "Hello";

// Output string value
cout << greeting;
```

C++ User Input

You have already learned that cout is used to output (print) values. Now we will use cin to get user input. cin is a predefined variable that reads data from the keyboard with the extraction operator (>>). cout is pronounced "see-out". Used for **output**, and uses the insertion operator (<<). cin is pronounced "see-in". Used for **input**, and uses the extraction operator (>>).

In the following example, the user can input a number, which is stored in the variable x. Then we print the value of x:

Example

```
int x;
cout << "Type a number: "; // Type a number and press enter
cin >> x; // Get user input from the keyboard
cout << "Your number is: " << x; // Display the input value
```

C++ Switch Statements

Same as Java Switch Statement

Example

```
int day = 4;
switch (day) {
  case 1:
    cout << "Monday";
    break;
  case 2:
    cout << "Tuesday";
    break;
  case 3:
    cout << "Wednesday";
    break;
  case 4:
    cout << "Thursday";
    break;
  case 5:
    cout << "Friday";
    break;
  case 6:
    cout << "Saturday";
    break;
  case 7:
    cout << "Sunday";
    break;
}
// Outputs "Thursday" (day 4)
```

C++ Loops

Same as Java loops.

Do loop Example

```
int i = 0;
while (i < 5) {
  cout << i << "\n";
  i++;
}
```

Do While loop Example

```
int i = 0;
do {
  cout << i << "\n";
  i++;
}
while (i < 5);
```

For loop Example

```
for (int i = 0; i < 5; i++) {
  cout << i << "\n";
}
```

C++ Arrays

Again, same as Java loops except no need of new operator.

```
int myNum[3] = {10, 20, 30};

string cars[4] = {"Toyota", "BMW", "Ford", "Mazda"};

cars[0] = "Suzuki";
cout << cars[0];

for (int i = 0; i < 3; i++) {

      cin>>myNum[i];    }
```

You don't have to specify the size of the array. But if you don't, it will only be as big as the elements that are inserted into it:

```
string cars[] = {"Volvo", "BMW", "Ford"}; // size of array is always 3
```

C++ Functions

C++ provides some pre-defined functions, such as main(), which is used to execute code. But you can also create your own functions to perform certain actions.

To create (often referred to as *declare*) a function, specify the name of the function, followed by parentheses **()**:

```
void myFunction() {
  // code to be executed
}
```

Call a Function

Declared functions are not executed immediately. They are "saved for later use", and will be executed later, when they are called.

To call a function, write the function's name followed by two parentheses () and a semicolon ;

In the following example, myFunction() is used to print a text (the action), when it is called:

Example

```
// Create a function
void myFunction() {
  cout << "I just got executed!";
}

int main() {
  myFunction(); // call the function
  return 0;
}
```

If the function is small then you may write it above the main() method but if it is large function then you have to define it after main() method.

```
// using function definition after main() function
// function prototype is declared before main()

#include <iostream>

using namespace std;

// function prototype
int add (int, int);
int subtract (int, int);
int multiply (int, int);

int main() {
    int sum, sub, mul;

    // calling the function and storing
    // the returned value in sum
    sum = add(100, 78);
    sub = subtract (100, 78);
    mul = multiply (10, 7);

    cout << "100 + 70 = " << sum << endl;
    cout << "100 - 70 = " << sub << endl;
    cout << "10 * 7 = " << mul << endl;

    return 0;
}
```

C++ Library Functions

Library functions are the built-in functions in C++ programming. Programmers can use library functions by invoking the functions directly; they don't need to write the functions themselves.

Some common library functions in C++ are sqrt(), abs(), isdigit(), etc.

In order to use library functions, we usually need to include the header file in which these library functions are defined. For instance, in order to use mathematical functions such as sqrt() and abs(), we need to include the header file cmath.

Structures in C++

Structure is a collection of variables of different data types under a single name. It is similar to a class in that, both holds a collecion of data of different data types.

For example: You want to store some information about a person: his/her name, citizenship number and salary. You can easily create different variables name, citNo, salary to store these information separately.

However, in the future, you would want to store information about multiple persons. Now, you'd need to create different variables for each information per person: name1, citNo1, salary1, name2, citNo2, salary2

You can easily visualize how big and messy the code would look. Also, since no relation between the variables (information) would exist, it's going to be a daunting task.

A better approach will be to have a collection of all related information under a single name Person, and use it for every person. Now, the code looks much cleaner, readable and efficient as well.

This collection of all related information under a single name Person is a structure.

The struct keyword defines a structure type followed by an identifier (name of the structure).

Then inside the curly braces, you can declare one or more members (declare variables inside curly braces) of that structure. For example:

```
struct Person
{
    char name[50];
    int age;
    float salary;
};
```

Here a structure person is defined which has three members: name, age and salary

When a structure is created, no memory is allocated.

The structure definition is only the blueprint for the creating of variables. You can imagine it as a datatype. When you define an integer as below:

```
int foo;
```

The int specifies that, variable foo can hold integer element only. Similarly, structure definition only specifies that, what property a structure variable holds when it is defined.

Note: Remember to end the declaration with a semicolon **(;)**

Once you declare a structure person as above. You can define a structure variable as:

```
Person bill;
```

Here, a structure variable bill is defined which is of type structure Person.

When structure variable is defined, only then the required memory is allocated by the compiler.

Considering you have either 32-bit or 64-bit system, the memory of float is 4 bytes, memory of int is 4 bytes and memory of char is 1 byte.
Hence, 58 bytes of memory is allocated for structure variable bill.

How to access members of a structure?

The members of structure variable is accessed using a **dot (.)** operator.
Suppose, you want to access age of structure variable bill and assign it 50 to it. You can perform this task by using following code below:

```
bill.age = 50;
```

Example: C++ Structure

C++ Program to assign data to members of a structure variable and display it.

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

struct Person
{
    char name[50];
    int age;
    float salary;
};

int main()
{
    Person p1;

    cout << "Enter Full name: ";
    cin.get(p1.name, 50);
    cout << "Enter age: ";
    cin >> p1.age;
    cout << "Enter salary: ";
    cin >> p1.salary;

    cout << "\nDisplaying Information." << endl;
    cout << "Name: " << p1.name << endl;
    cout <<"Age: " << p1.age << endl;
    cout << "Salary: " << p1.salary;

    return 0;
}
```

Output

```
Enter Full name: Abdul Rashid
Enter age: 27
Enter salary: 100000

Displaying Information.
Name: Abdul Rashid
Age: 27
Salary: 100000
```

Here a structure Person is declared which has three members name, age and salary. Inside main() function, a structure variable p1 is defined. Then, the user is asked to enter information and data entered by user is displayed.

Lab Task:

C++ program to include all the topics (from data types to structures).