Lab-01

**Introduction to C++ and Review of Basic Programming Concepts**

1. **Objectives :**

Objective of this lab is to review basic programming concepts, understanding C++ syntax, getting familiar with the concepts of structures and usage of Code Blocks IDE.

1. **Outcomes :**
   1. The student will be able to understand difference between C and C++ syntax.
   2. The student will be able to write simple codes using Code Blocks IDE.
   3. The student will be able to declare and use structures.
   4. The student will be able to pass arguments to functions both by value and by reference.
2. **Introduction:**

C++ is a general purpose programming language that supports various computer programming models such as object-oriented programming and generic programming. It was created by Bjarne Stroustrup and,

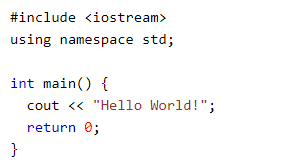
***“Its main purpose was to make writing good programs easier and more pleasant for the individual programmer.”***

By learning C++, you can create applications that will run on a wide variety of hardware platforms such as personal computers running Windows, Linux, UNIX, and Mac OS X, as well as small form factor hardware such as IoT devices like the Raspberry PI and Arduino–based boards.

C++ is derived from the C-Language. Almost every correct statement in C is also correct in C++, although the reverse is not true.

* 1. **Syntax of C++:**

**Basic program:**



**Explanation:**

**Line 1:** **#include <iostream>** is a header file library that lets us work with input and output objects, such as cout (used in line 5). Header files add functionality to C++ programs.

**Line 2:** **using namespace std** means that we can use names for objects and variables from the standard library.

**Line 3:** A blank line. C++ ignores white space.

**Line 4:** Another thing that always appear in a C++ program, is **int main()**. This is called a function. Any code inside its **curly brackets {}** will be executed.

**Line 5:** **cout** (pronounced "see-out") is an object used together with the **insertion operator (<<)** to output/print text. In our example it will output "Hello World".

**Note:** Every C++ statement ends with a semicolon ; .

**Line 6:** **return 0** ends the main function.

* 1. **Difference between C and C++:**

The **main difference** between both these languages is C is a procedural programming language and does not support classes and objects, while C++ is a combination of both procedural and object-oriented programming languages.

In **procedural programming**, each statement in the language tells the computer to do something: get some input, add these numbers divide by six, display that output. A program in procedural language is a list of instructions.

For every small program, no other organizing principle is needed. The programmer creates the list of instructions and the computer carries them out.

In **object – oriented programming** is to combine into a single unit both data and the function that operate on that data. Such a unit is called an object. Keep in mind that object – oriented programming is not primary concerned with the details of program operation, instead it deals with the overall organization of the program.

|  |  |  |
| --- | --- | --- |
| **Feature** | **C** | **C++** |
| Language Type | As mentioned before C is procedural programming. | On the other hand, C++ supports both procedural and object-oriented programming paradigms. |
| OOPs feature Support | As C does not support the OOPs concept so it has no support for polymorphism, encapsulation, and inheritance. | C++ has support for polymorphism, encapsulation, and inheritance as it is being an object-oriented programming language |
| Data Security | As C does not support encapsulation so data behave as a free entity and can be manipulated by outside code. | On another hand in the case of C++ encapsulation hides the data to ensure that data structures and operators are used as intended. |
| Driven type | C in general known as function-driven language. | On the other hand, C++ is known as object driven language. |
| Feature supported | C does not support function and operator overloading also do not have namespace feature and reference variable functionality. | On the other hand, C++ supports both function and operator overloading also have namespace feature and reference variable functionality. |

* 1. **Difference between Syntax of C and C++:**
* **Namespace** is used by C++, which avoid name collisions.
* **Header file** stdio.h in C, iostream.h in C++
* **scanf() and printf()** functions are used for input/output in C, **cin and cout** are used for input/output in C++.
  1. **Structure:**

A structure is a user defined data type. Through structures you have the ability to define a new type of data considerably more complex than the types we have been using. A structure is a combination of several different data types. It is similar to a class in that, both holds a collecion of data of different data types. It is declared by using the keyword struct followed by the structure name.

**Syntax:**

struct struct\_name

{

Data\_type1 member\_name1;

Data\_type2 member\_name2;

Data\_type3 member\_name3;

} object\_name;

* 1. **Passing arguments by Value:**

By definition, pass by value means you are **making a copy in memory of the actual parameter's value that is passed** in, a copy of the contents of the actual parameter.

A parameter passing mechanism in which the value of actual parameter is copied to formal parameters of called functions is known as pass by value. If the function makes any change in formal parameter, it does not affect the values of actual parameter. **It is the default mechanism for passing parameters to functions.**

**Actual Parameters** are the values that are passed to the function when it is invoked while **Formal Parameters** are the variables defined by the function that receives values when the function is called.

* 1. **Passing arguments by Reference:**

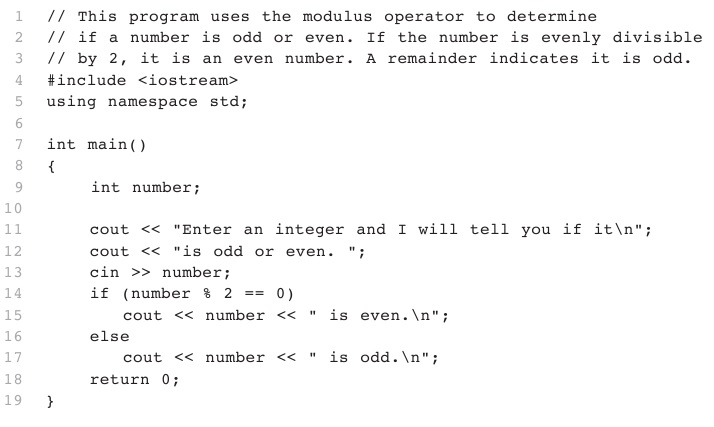
A parameter passing mechanism in which the address of actual parameter is passed to the called function is known as **pass by reference (also called pass by address**). In pass by reference, we declare the function parameters as references rather than normal variables. The formal parameter is not created separately in the memory. Formal parameter becomes a second name of actual parameter. It means that single memory is shared between actual parameter and formal parameter. If the called function makes any change in formal parameter, the change is also visible in actual parameter.

**Difference:**

|  |  |
| --- | --- |
| **Pass by value** | **Pass by reference** |
| A copy of variable is created and changes are not reflected in original memory address | Instead of creating a copy of variable changes are made at original memory address |

1. **Examples:**

**Example-1:**

****

**Example-2:**

**Write a program to store a person’s name using char array, age in integer numbers and salary in float numbers. Use structure to store this information.**

#include <iostream>

using namespace std;

struct Person

{

char name[50];

int age;

float salary;

};

int main()

{

Person p1;

cout << "Enter Full name: ";

cin>>p1.name;

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;

}

**Example-3:**

**A simple example to show how arguments are passed by value to a function.**

#include <iostream>

using namespace std;

void duplicate (int var1, int var2, int var3) //formal parameters

{

var1\*=2; //change value of both just formal parameter

var2\*=2;

var3\*=2;

//cout<< "x=" << var1 << ", y=" << var2 << ", z=" << var3;

}

int main ( )

{

int x=1, y=3, z=7; //actual parameters

duplicate (x, y, z); // pass actual parameters to function

cout<<endl;

cout<< "x=" << x << ", y=" << y << ", z=" << z;

return 0;

}

**Example-4:**

**An example to demonstrate how arguments are passed by reference**

#include <iostream>

using namespace std;

void duplicate (int& var1, int& var2, int& var3) //formal parameters

{

var1\*=2; //change value of both actual and formal parameter (var1=var1\*2)

var2\*=2;

var3\*=2;

cout<< "x=" << var1 << ", y=" << var2 << ", z=" << var3;

}

int main ( )

{

int x=1, y=3, z=7; //actual parameters

duplicate (x, y, z); // pass actual parameters to function

cout<<endl;

cout<< "x=" << x << ", y=" << y << ", z=" << z;

return 0;

}

**OUTPUT:**

x=2, y=6, z=14

1. **In-Lab Tasks:**
   1. Write a program that declares a structure to store date. Declare an instance of this structure to represent date of birth. The program should read the day, month and year values of birth date and display date of birth in dd/mm/yy format.
   2. Write a program that declares a structure to store Student data containing his name , age and Roll#. Use array of structures to represent record of 3 students.
   3. Write a program that declares a structure to store book name, price and pages of a book. The structure should include functions to assign user defined values to each book and display the record of most costly book.
   4. Write a function that swaps the values of two integer variables
      1. using pass by value
      2. and pass by reference and see their differences
2. **Post-Lab Tasks:** 
   1. There is a structure called **employee** that holds information like employee code, name, date of joining. Write a program to create an array of the structure and enter some data into it. Then ask the user to enter current date. Display the names of those employees whose tenure is 3 or more than 3 years according to the given current date