C++ CLASSES AND OBJECTS

The main purpose of C++ programming is to add object orientation to the C programming language and classes are the central feature of C++ that supports object-oriented programming and are often called user-defined types.

A class is used to specify the form of an object and it combines data representation and methods for manipulating that data into one neat package. The data and functions within a class are called members of the class.

C++ Class Definitions:

When you define a class, you define a blueprint for a data type. This doesn't actually define any data, but it does define what the class name means, that is, what an object of the class will consist of and what operations can be performed on such an object.

A class definition starts with the keyword **class** followed by the class name; and the class body, enclosed by a pair of curly braces. A class definition must be followed either by a semicolon or a list of declarations. For example, we defined the Box data type using the keyword **class** as follows:

```
class Box
{
   public:
      double length;     // Length of a box
      double breadth;    // Breadth of a box
      double height;     // Height of a box
};
```

The keyword **public** determines the access attributes of the members of the class that follow it. A public member can be accessed from outside the class anywhere within the scope of the class object. You can also specify the members of a class as **private** or **protected** which we will discuss in a sub-section.

Define C++ Objects:

A class provides the blueprints for objects, so basically an object is created from a class. We declare objects of a class with exactly the same sort of declaration that we declare variables of basic types. Following statements declare two objects of class Box:

```
Box Box1;          // Declare Box1 of type Box
Box Box2;          // Declare Box2 of type Box
```

Both of the objects Box1 and Box2 will have their own copy of data members.

Accessing the Data Members:

The public data members of objects of a class can be accessed using the direct member access operator . . Let us try the following example to make the things clear:

```
#include <iostream>

using namespace std;

class Box

{
```

```
{
   Box Box1;        // Declare Box1 of type Box  Box Box2;   // Declare Box2 of type
   Box
   double volume = 0.0;     // Store the volume of a box here

   // box 1 specification  Box1.height = 5.0;
   Box1.length = 6.0;
   Box1.breadth = 7.0;

   // box 2 specification  Box2.height = 10.0;
   Box2.length = 12.0;
   Box2.breadth = 13.0;
   // volume of box 1
   volume = Box1.height * Box1.length * Box1.breadth;  cout << "Volume of Box1 : " <<
   volume <<endl;

   // volume of box 2
   volume = Box2.height * Box2.length * Box2.breadth;  cout << "Volume of Box2 : " <<
   volume <<endl;  return 0;
}
```

When the above code is compiled and executed, it produces the following result:

```
Volume of Box1 : 210  Volume of Box2 : 1560
```

It is important to note that private and protected members can not be accessed directly using direct member access operator . . We will learn how private and protected members can be accessed.

Classes & Objects in Detail:

So far, you have got very basic idea about C++ Classes and Objects. There are further interesting concepts related to C++ Classes and Objects which we will discuss in various sub-sections listed below:

Concept	Description
Class member functions	A member function of a class is a function that has its definition or its prototype within the class definition like any other variable.
Class access modifiers	A class member can be defined as public, private or protected. By default members would be assumed as private.
Constructor & destructor	A class constructor is a special function in a class that is called when a new object of the class is created. A destructor is also a special function which is called when created object is deleted.
C++ copy constructor	The copy constructor is a constructor which creates an object by initializing it with an object of the same class, which has been created previously.
C++ friend functions	A **friend** function is permitted full accessto private and protected members of a class.
C++ inline functions	With an inline function, the compiler tries to expand the code in the body of the function in place of a call to the function.

The this pointer in C++	Every object has a special pointer **this** which points to the object itself.
Pointer to C++ classes	A pointer to a class is done exactly the same way a pointer to a structure is. In fact a class is really just a structure with functions in it.
Static members of a class	Both data members and function members of a class can be declared as static.

"Class"

* Class :

A class in c++ is the building block, that leads to Object-oriented programming. It is user-defined data type, which holds its own data members and member functions which can be accessed and used by creating an instance of class.

Syntax:

```
class className
{
    // some data
    // some functions
};
```

* Three Access Specifiers :

→ **Public:** The public access specifier allows a class to object subject its member variable and member functions to other function and object.

→ **Private:** The private access specifier allow a class to hide its member variable and member function from other class objects and function.

(Default Access specifier)

→ **Protected:** The protected access specifier allows a class to hide its member variables and member function from other class objects and function just like private access specifier - is used while implenting inheritance.

```
class class_name
{
    private
    ...
    ...
    ...
    public
    ...
    ...
    ...
};
```

private members or method

public members or method

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

class Employee
{
    private:
        int a,b,c;
    public:
        int d,e;
```

```
void setdata(int a1, int b1, int c1);
void getdata(){

    Cout<<"The value of a is "<<a<<endl;
    Cout<<"the value of b " "<<b<<endl;
    Cout<<"  "  "  " " c " "<<c<<endl;
    Cout<<"  "  "  " " d " "<<d<<endl;
    Cout<<"  "  "  " " e " "<<e<<endl;
}
};

void Employee :: setdata(int a1, int b1, int c1){

    a = a1;
    b = b1;
    c = c1;
}

int main(){
    Employee harry;

harry.d = 34;
harry.e = 89;
harry.setdata(1,2,3);
harry.getdata();
return 0;
}
```