**Lab 03 – *Class Scope and Accessing Class Members***

1. **Objectives**

The objective of this lab is to teach the students, the scope of class data members and its member functions.

1. **Outcome**

At the end of this lab student will be familiar with the accessing rules of class data members and member functions

1. **Introduction**

In object oriented programming, methods and variables have various scope. Scope means that the method or variable may or may not be directly accessible to other objects or classes. Classes that do not have instances may be accessible to the system.

One of the techniques in object-oriented programming is encapsulation. It concerns the hiding of data in a class and making them available only through its methods. In this way the chance of making accidental mistakes in changing values is minimized. C++ allows you to control access to classes, methods, and fields via so-called access modifiers. The access to classes, constructors, methods and fields are regulated using access modifiers i.e. a class can control what information or data can be accessible by other classes. To take advantage of encapsulation, you should minimize access whenever possible.

* 1. **Class Scope**

Class variables and class methods are associated with a class. An instance of the class (object) is not required to use these variables or methods. Class methods cannot access instance variables or methods, only class variables and methods.

* 1. **Instance Scope**

Instance variables and instance methods are associated with a specific object. They can access class variables and methods.

* 1. **Private Scope**

A **private** member variable or function cannot be accessed, or even viewed from outside the class. Only the class and friend functions can access private members. Private variables and private methods are only accessible to the object they are contained in.

* 1. **Protected Scope**

Protected variables and protected methods are accessible by the class they are in and inheriting classes (sub classes) only.

* 1. **Public Scope**

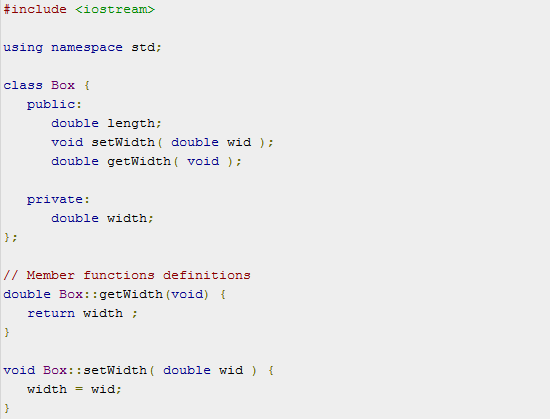
A **public** member is accessible from anywhere outside the class but within a program. You can set and get the value of public variables without any member function

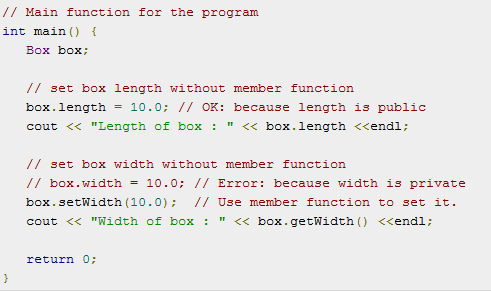
* 1. **Encapsulation**

The process of providing a public interface to interact with the object while hiding other information inside the object is called encapsulation.

1. **Examples**

**Example #01**





**Example #02**

The following program illustrates the usage of objects and classes in C++:

//The program uses public member functions to input two private numbers, add two private numbers and display the result

#include<iostream>

using namespace std;

class add //Specifies the class

{ private:

int iNum1, iNum2, iNum3; //Member data

public:

void input(int iVar1, int iVar2) //Member function

{

cout<<"Functions to assign values to the member data"<<endl; iNum1=iVar1; iNum2=iVar2;

}

void sum(void) //Member function

{

cout<<"Functions to find the sum of two numbers"<<endl;

iNum3=iNum1+iNum2;

}

void disp(void) //Member function

{

cout<<"The sum of the two numbers is "<<iNum3<<endl;

}

};

/////////main function of the program/////////// void main()

{

add A1; // class name and object name

int iX, iY;

cout<<"Input two numbers"<<endl; cin>>iX; cin>>iY;

A1.input(iX, iY);

A1.sum();

A1.disp();

}

1. **In Lab Tasks** 
   1. Code the example given above and check the errors if you try to access the private data members in main() function. Also modify the above task by making the scope of public member functions as private. Create access functions in public scope to access private member functions from main().
   2. Create an employee class, The member data should comprise an int for storing the employee number and a float for storing the employee’s compensation. Member functions should allow the user to enter this data and display it. Write a main() that allows the user to enter data for three employees and display it.
   3. Create a class called time that has separate int member data for hours, minutes, and  
      seconds. One constructor should initialize this data to 0, and another should initialize it  
      to fixed values. Another member function should display it, in 11:59:59 format. The final member function should add two objects of type time passed as arguments.  
      A main() program should create two initialized time objects and  
      one that isn’t initialized. Then it should add the two initialized values together, leaving the result in the third time variable. Finally it should display the value of this third variable.
   4. Create a fraction class. Member data is the fraction’s numerator and denominator. Member functions should accept input from the user in the form 3/5, and output the fraction’s value in the same format. Another member function should add two fraction values. Write a main() program that allows the user to repeatedly input two fractions and then displays their sum. After each operation, ask whether the user wants to continue.
2. **Post Lab Tasks** 
   1. Create a date class. Its member data should consist of three ints: month, day, and year. It should also have two member functions: getdate() , which allows the user to enter a date in 12/31/02 format, and showdate() , which displays the date.
   2. Create a class of subtraction having two private data members. Create member functions to get data from users and for subtraction of data members. Use appropriate access modifiers for class methods.