**Logo, company name

Description automatically generated**

**Department of (Computer Science)**

**Pak-Austria** Fachhochschule**: Institute of Applied Sciences and Technology, Haripur, Pakistan**

**COMP-112L Object Oriented Programming Lab**

**Lab Journal**

**Class: BS Computer Science**

**Name: Ahmed Raza**

**Registration No.: B20F0436CS031**

**Semester: 3rd**

**Submission Date: 08 February, 2022**

**Submitted to: Engr. Rafi-Ullah**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Instructor Signature**

**Lab No. 07**

**Friend Function & Classes, This Pointer, and static Variables**

**Objectives:**

In this lab we will be discussing about Friend Function & Classes, This Pointer, and static Variables in detail. This is one of the most important concepts in Object Orientation C++. C++ provides the friend keyword to do just this. Inside a class, you can indicate that other classes (or simply functions) will have direct access to protected and private members of the class.

**Tools/Software Required:**

* All the tasks are implemented on DEV C++.

**Introduction:**

**Friend Classes**

We must specify that the access is granted for a class using the class keyword:

friend class aClass;

Friend declarations can go in either the public, private, or protected section of a class--it doesn't matter where they appear. In particular, specifying a friend in the section marked protected doesn't prevent the friend from also accessing private fields.

**Friend Functions**

“Friend functions are not members of the class, and therefore the access attributes do not apply to them. They are just ordinary global functions with special privileges”.

A class can grant access to its internal variables on a more selective basis--for instance, restricting access to only a single function. To do so, the entire function signature must be replicated after the friend specifier, including the return type of the function--and, of course, you'll need to give the scope of the function if it's inside another class:

**friend return\_type class\_name::function(args);**

When friends are specified within a class, this does not give the class itself access to the friend function. That function is not within the scope of the class; it's only an indication that the class will grant access to the function.

Functions which are friends of a class and are defined within the class definition are also by default inline.

**Placing friend Function Definitions Inside the** **Class**

We could have combined the definition of the function with its declaration as a friend of the Box class within the class definition - the code would run as before. However, this has a number of disadvantages relating to the readability of the code. Although the function would still have global scope, this wouldn't be obvious to readers of the code, since the function would be hidden in the body of the class definition, and particularly since the function would no longer show up in the Class View.

**The this pointer**

Every object in C++ has access to its own address through an important pointer called this pointer. The **this** pointer is an implicit parameter to all member functions. Therefore, inside a member function, this may be used to refer to the invoking object.

Friend functions do not have a **this** pointer, because friends are not members of a class. Only member functions have a **this** pointer.

**Static keyword in C++**

The static keyword has another meaning when applied to global variables — it changes them from global scope to file scope. Because global variables are typically avoided by competent programmers, and file scope variables are just global variables limited to a single file, the static keyword is typically not used in this capacity.

**Static member variables**

When we instantiate a class object, each object gets it’s own copy of all normal member variables. Member variables of a class can be made static by using the static keyword. Static member variables only exist once in a program regardless of how many class objects are defined! One way to think about it is that all objects of a class-share the static variables.

**Lab Tasks:**

**Task # 01:**

Define a class **Addition** which has **static** two member variables **num1, num2** and one **static** member function **add.** This program return the resultant sum of two numbers.

**Code:**

**#include <iostream>**

**using namespace std;**

**class Addition**

**{**

**private:**

**static int a;**

**static int b;**

**public:**

**static int sum();**

**void set\_a(int num1)**

**{**

**this->a=num1;**

**}**

**void set\_b(int num2)**

**{**

**this->b=num2;**

**}**

**int get\_a(void)**

**{**

**return a;**

**}**

**int get\_b(void)**

**{**

**return b;**

**}**

**};**

**int Addition::a=0;**

**int Addition::b=0;**

**int Addition::sum()**

**{**

**return (a+b);**

**};**

**int main()**

**{**

**Addition A;**

**int x,y;**

**cout<<"Enter the first number: ";**

**cin>>x;**

**cout<<"Enter the seocond number: ";**

**cin>>y;**

**A.set\_a(x);**

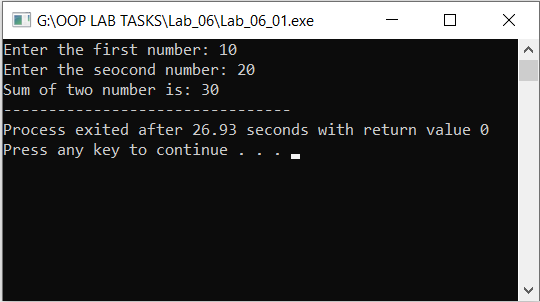
**A.set\_b(y);**

**cout<<"Sum of two number is: "<<A.sum();**

**return 0;**

**}**

**Output:**

****

**Task #02:**

Define a class **CalculateArea** having three data members length, breadth, area and another **friend** class **Area** that print the calculated area. The data members should be initialized through constructor.

**Code:**

**#include <iostream>**

**using namespace std;**

**class CalculateArea**

**{**

**private:**

**double length;**

**double breadth;**

**double area;**

**public:**

**CalculateArea(double lv=1.0, double bv=1.0, double ar=0)**

**{**

**cout << endl << "Constructor called.";**

**length = lv;**

**breadth = bv;**

**area = ar;**

**}**

**double Area()**

**{**

**cout<<endl<<"Enter length: ";**

**cin>>length;**

**cout<<"Enter breadth: ";**

**cin>>breadth;**

**return area = length \* breadth ;**

**}**

**friend double CalculateAreaSurface(CalculateArea aCalculateArea);**

**};**

**double CalculateAreaSurface(CalculateArea aCalculateArea)**

**{**

**return (aCalculateArea.length\*aCalculateArea.breadth );**

**}**

**int main(void)**

**{**

**CalculateArea aCalculateArea(0,0);**

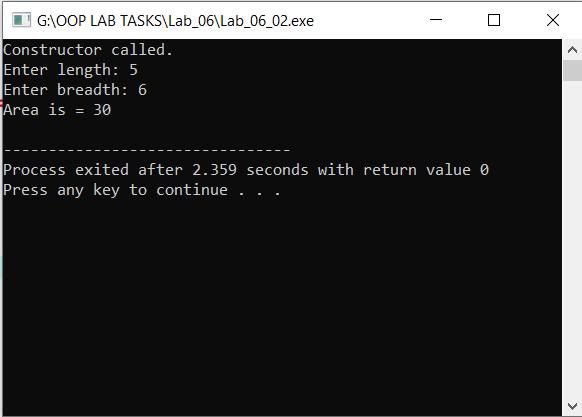
**cout << "Area is = "<< aCalculateArea.Area();**

**cout <<endl;**

**return 0;**

**}**

**Output:**

****

**Results & Observations:**

In this Lab I’ve learned about the concept of Friend Class & Static Variables & also I understand the concept of friend function that how can we use friend function in the class. As well as I learned about static keywords and static member variables. In the first task, I’ve used a Class with name Additionwhich has static two member variables num1**,** num2 and one static member function add and In the second task I’ve used class and constructer, also I’ve use a Area function to calculate area then I’ve used Friend function in the class through which we can access private data.