A picture containing text

Description automatically generatedA white text on a black background

AI-generated content may be incorrect.

Object Oriented Programming Lab (Spring-2025)

Teacher: Engr. Muhammad Faisal Zia

Department of Software Engineering

Bahria University, Islamabad

Student: Ahtisham Khan

Enrollment: 09-131242-014

Lab Journal: 4

Date: 15/03/2025

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task No:** | **Task Wise Marks** | | **Documentation Marks** | | | **Total Marks**  **(20)** |
| **Assigned** | **Obtained** | **Assigned** | **Obtained** | |
| 1 | 2 |  | 4 |  | A colorful lines on a black background  AI-generated content may be incorrect. | |
| 2 | 2 |  |
| 3 | 2 |  |
| 4 | 5 |  |
| 5 | 5 |  |

**Comments:**

**Signature**

**Lab # 5**

**Friend Function And Friend Class**

**Objective:**

The objective of this lab is to understand the concept of **friend functions** and **friend classes** in C++. By the end of this lab, you should be able to:

1. Define and use friend functions to access private and protected members of a class.
2. Define and use friend classes to access private and protected members of another class.
3. Apply friend functions and friend classes in real-world scenarios like mathematical operations and factorial calculations.

One of the important concepts of OOP is data hiding, i.e., a non member function cannot access an object's private or protected data. But, sometimes this restriction may force programmers to write long and complex codes. So, there is a mechanism built in C++ programming to access private or protected data from non-member functions. This is done using a friend function or/and a friend class.

**1. Friend Functions**

* A friend function is a non-member function that has access to the private and protected members of a class.
* It is declared inside the class using the **friend** keyword.
* It is defined outside the class like a normal function.
* The compiler knows a given function is a friend function by the use of the keyword friend. For accessing the data, the declaration of a friend function should be made inside the body of the class (can be anywhere inside class either in private or public section) starting with keyword friend. Declaration of friend function in C++

**Syntax**

A black rectangular object with white text

AI-generated content may be incorrect.

**Example: Working of a Friend Function**

/\* C++ program to demonstrate the working of friend function.\*/

#include <iostream>

using namespace std;

class Distance {

private:

int meter;

public:

Distance() : meter(0) {} // Constructor to initialize meter to 0

// Declare friend function

friend int addFive(Distance);

};

// Friend function definition

int addFive(Distance d) {

d.meter += 5; // Access private data member

return d.meter;

}

int main() {

Distance D;

cout << "Distance: " << addFive(D) << endl;

A black rectangle with white and blue text

AI-generated content may be incorrect.

return 0;

}

**Example: Add member of two different Classes**

// Add members of two different classes using friend functions

#include <iostream>

using namespace std;

// forward declaration

class ClassB;

class ClassA {

public:

// constructor to initialize numA to 12

ClassA() : numA(12) {}

private:

int numA;

// friend function declaration

friend int add(ClassA, ClassB);

};

class ClassB {

public:

// constructor to initialize numB to 1

ClassB() : numB(1) {}

private:

int numB;

// friend function declaration

friend int add(ClassA, ClassB);

};

// access members of both classes

int add(ClassA objectA, ClassB objectB) {

return (objectA.numA + objectB.numB);

}

int main() {

ClassA objectA;

A black rectangle with white numbers

AI-generated content may be incorrect. ClassB objectB;

cout << "Sum: " << add(objectA, objectB);

return 0;

}

One thing to notice here is the friend function inside ClassA is using the ClassB. However, we haven't defined ClassB at this point.

// inside classA

friend int add(ClassA, ClassB);

For this to work, we need a forward declaration of ClassB in our program.

// forward declaration

class ClassB;

**2. Friend Class**

* A **friend class** is a class that has access to the private and protected members of another class.
* It is declared inside the class using the friend keyword.
* When a class is declared a friend class, all the member functions of the friend class become friend functions.
* Since ClassB is a friend class, we can access all members of ClassA from inside ClassB.
* *However, we cannot access members of ClassB from inside ClassA. It is because friend relation in C++ is only granted, not taken.*

**Syntax**

A computer screen with white text

AI-generated content may be incorrect.

**Example: C++ Friend Class**

// C++ program to demonstrate the working of friend class

#include <iostream>

using namespace std;

// forward declaration

class ClassB;

class ClassA {

private:

int numA;

// friend class declaration

friend class ClassB;

public:

// constructor to initialize numA to 12

ClassA() : numA(12) {}

};

class ClassB {

private:

int numB;

public:

// constructor to initialize numB to 1

ClassB() : numB(1) {}

// member function to add numA

// from ClassA and numB from ClassB

int add() {

ClassA objectA;

return objectA.numA + numB;

}

};

int main() {

ClassB objectB;

cout << "Sum: " << objectB.add();

return 0;

}

**Conclusion:**

In this lab, we learned:

1. How to define and use friend functions to access private and protected members of a class.
2. How to define and use friend classes to access private and protected members of another class.
3. Practical applications of friend functions and friend classes in mathematical operations and factorial calculations.

**Additional Notes:**

* Friend functions and friend classes violate the principle of data hiding, so they should be used sparingly and only when necessary.
* Friend functions are useful for operator overloading and other scenarios where direct access to private members is required.

**Lab Tasks/ Homework**

**Task 01:** Create two classes: alpha and beta (friend class). The constructors in these classes initialize their single data item to fixed values (3 in alpha and 7 in beta).

Create a friend function that have access to both private data member of these classes.

An object of each class is passed as an argument to the friend function. This friend function adds the data items and return the sum. The output of the program should be like:

Output should be like this

A screenshot of a computer code

AI-generated content may be incorrect.

**Solution:**

Enter your code here…

#include<iostream>

using namespace std;

class Alpha {

private:

int numA;

friend class Beta;

public:

Alpha() :numA(3) {

}

friend int add(Alpha, Beta);

};

class Beta {

private:

int numB;

public:

Beta() :numB(7) {

}

friend int add(Alpha, Beta);

};

int add(Alpha objA,Beta objB) {

return objA.numA+objB.numB;

}

int main() {

system("title Ahtisham khan (014) OOP LAB 5");

Alpha objA;

Beta objB;

cout << "Sum of Data Members: " <<add(objA,objB);

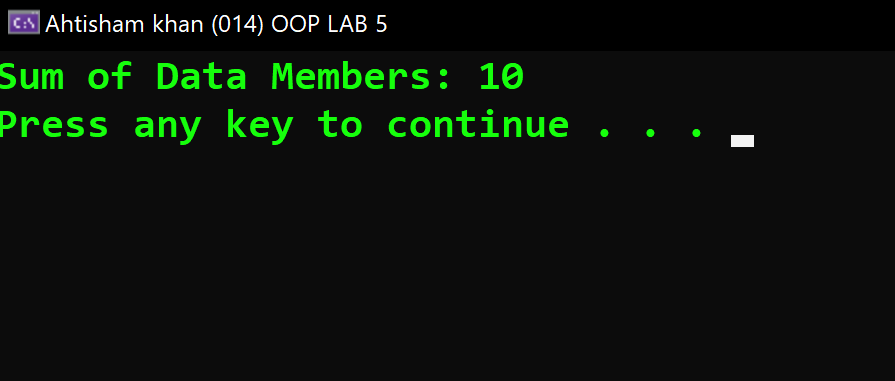
cout << endl;

system("pause");

}

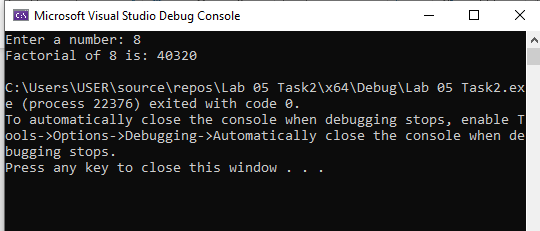
**Output:**

Your Output here…



In case of output snippet please make sure output snippet contains student name and id. AliAhmed\_123\_Lab05\_T1.exe

**Task 02:** Write a program to calculate the factorial of a number using a friend function. The number should be a private data member of a class. The output of the program should be like this



**Solution:**

Enter your code here…

#include<iostream>

using namespace std;

class CalFactorial {

private:

int num;

public:

CalFactorial() :num(0) {

}

CalFactorial(int n) {

num = n;

}

friend int calfact(CalFactorial);

};

int calfact(CalFactorial fa) {

int fact = 1;

for (int i = 1;i <=fa.num;i++) {

fact \*= i;

}

return fact;

}

int main() {

system("title Ahtisham khan (014) OOP LAB 5");

int a;

cout << "Enter a number: " << endl;

cin >> a;

CalFactorial fa(a);

cout << "Factorial of a number: " << calfact(fa) << endl;

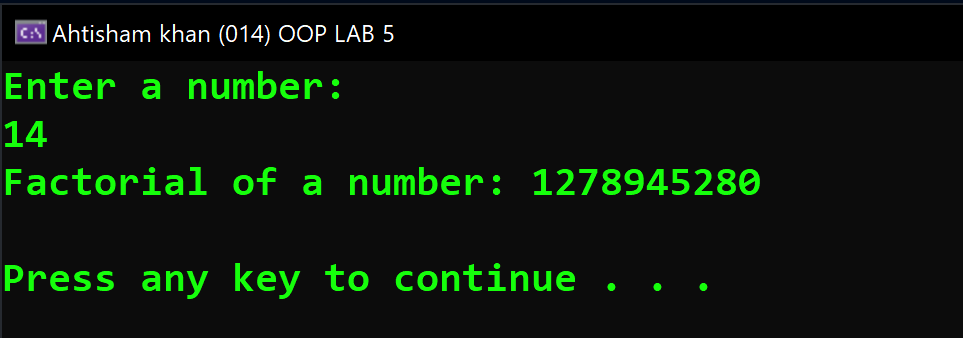
cout << endl;

system("pause");

}

**Output:**

Your Output here…



In case of output snippet please make sure output snippet contains student name and id. AliAhmed\_123\_Lab05\_T2.exe

**Task 03:** Write a program to display the sum, difference, product and division of two numbers using friend function. Create a class alpha. Declare four friend functions:

A screenshot of a computer

AI-generated content may be incorrect.

friend double sum (alpha)

friend double sub(alpha)

friend double mul(alpha )

friend double div(alpha )

Display the result. The output of the program should be like:

**Solution:**

Enter your code here…

#include<iostream>

using namespace std;

class Alpha {

private:

double num1;

double num2;

public:

//Default constructor

Alpha() :num1(0), num2(0) {

}

//parameterized constructor

Alpha(double a, double b) {

num1 = a;

num2 = b;

}

//Defining friend functions

friend double calsum(Alpha);

friend double calsab(Alpha);

friend double calmultiplication(Alpha);

friend double caldivision(Alpha);

};

//global function to calculate sum

double calsum(Alpha four) {

double sum;

sum = four.num1 + four.num2;

return sum;

}

//global function to calculate sabstraction

double calsab(Alpha four) {

double sab;

sab = four.num1 - four.num2;

return sab;

}

//global function to calculate multiplication

double calmultiplication(Alpha four) {

double multi;

multi = four.num1 \* four.num2;

return multi;

}

//global function to calculate Division

double caldivision(Alpha four) {

double division;

if (four.num2 == 0) {

cout << "Enter valid input" << endl;

return 0;

}

else

{

division = four.num1 / four.num2;

}

return division;

}

int main() {

system("title Ahtisham khan (014) OOP LAB 5");

double c, d;

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

cin >> c;

cout << "Enter the second number: " << endl;

cin >> d;

Alpha four(c,d);

cout << "Sum= " << calsum(four)<<endl;

cout << "Sabstraction= " << calsab(four)<<endl;

cout << "Multiplication= " << calmultiplication(four) << endl;

cout << "Division= " << caldivision(four) << endl;

cout << endl;

system("pause");

}

**Output:**

Your Output here…

