# Lab 3 Instructor: Saroj Shakya

Friend Functions

The friend function is **not a member** of any classes (non member function)

To make an outside function friend to a class, **declare this function as a friend** of the class by declaring the function preceded by the keyword friend e.g.

class A{

public:

friend void xyz();

};

The function is defined elsewhere in the program but the function definition **does not use** either the **keyword friend** or the **scope operator ::**

A friend function, although **not a member function**, **has full access rights to private member**s of the class

A friend function is **invoked like a normal function** without the help of any object

**A program to check if a number is even or odd using a friend function**

**class Number{**

**int num;**

**public:**

**void setNum(int x){**

**num = x ;**

**}**

**friend void checkNum(Number first); //friend function declaration**

**};**

**void checkNum(Number first){ //friend function definition**

**if(first.num %2 ==0) //friend function accessing private data**

**cout <<”Even Number”;**

**else**

**cout <<”Odd Number”;**

**}**

**void main(){**

**Number first;**

**first.setNum(10);**

**checkNum(first); //calling friend function**

**}**

Characteristics of a Friend function:

- It is **not in the scope of the class** to which it has been declared as friend (i.e. it is a non member function of a class)

- Since it not in the scope of the class it **cannot be called using the object of that class**

- It does not require an object for invoking it, it can be **invoked like a normal function** without the help of any object

- The member functions of a class can access the member functions and data members directly, BUT a friend function cannot access the member names directly and has to **use a object name and dot membership operator** with each member name e.g. A.x (for this purpose a friend function normally **takes objects as arguments**)

- It can be **declared either in the public or private section** of a class without affecting its meaning

- Usually it has objects as arguments

- Friend functions are **often used in operator overloading**

F**orward Declaration**, is required to let the compiler know that there is a definition of the class two coming up , otherwise the compiler would generate an error seeing the name of ‘two’ class in the declaration of the friend function friend void **friend void findMax(one o, two t);** in the One class below.

**Use of friend function to find the maximum value**

|  |  |
| --- | --- |
| **class two; //forward declaration**  **class one{**  **int a;**  **public:**  **void setA(int x){**  **a=x ;**  **}**  **void showA(){**  **cout<<a;**  **}**  **friend void findMax(one o, two t);**  **//declaration of friend function**  **};**  **class two{**  **int b;**  **public:**  **void setB(int y){**  **b=y ;**  **}**  **void showB(){**  **cout<<b;**  **}**  **friend void findMax(one o, two t);**  **//declaration of friend function**  **};** | **void findMax(one o, two t){ //definition of a friend function**  **if(o.a >t.b)**  **cout<<"max is "<<o.a<< " of class one";**  **else**  **cout<<"max is "<<t.b<< " of class two";**  **}**  **void main(){**  **one o; two t;**  **o.setA(300);**  **t.setB(200);**  **findMax(o,t);**  **}** |

Returning Objects from a Member Function

Just like a member function can receive an object as an argument it can also return an object

**Adding two numbers by returning an object**

**class Number{**

**int num;**

**public:**

**void setNum(int x){**

**num = x ;**

**}**

**void showNum(){**

**cout <<num;**

**}**

**friend Number addNum(Number first, Number second); //friend function declaration**

**};**

**Number addNum(Number first, Number second){ //friend function definition**

**Number temp; //declaring a temporary object temp**

**temp.num = first.num + second.num; //friend function accessing private data**

**return temp; //returning temp object**

**}**

**void main(){**

**Number first, second, third;**

**first.setNum(10);**

**second.setNum(11);**

**third = addNum(first, second); //friend function returns object**

**third.showNum();**

**}**

**Exercises:**

Create two classes Manager and Scientist,

1. Manager has a data member called mTax and Scientist has a data member called sTax to record the tax values
2. Include constructors to initialized these data member
3. Include a common friend function **friend void compute\_tax(Manager m, Scientist s);** to operate on objects of both these classes
4. Write full program to compare the taxes of objects of both these classes

Write a C++ program to create a class called **Point**

1. Include two data members x and y to the class
2. Include constructors to initialized the values of x and y
3. Add a member function to display the value of x and y
4. Add a friend function called friend void swap(point &a, point &b); that takes in two objects of type Point
5. Write a complete program to swap two points and display the swapped values

A program to add two complex numbers

A program to add two distances numbers