Object Oriented Programming in C++

Nepal College of Information Technology

Course Instructor: Er. Rabina Chaudhary

Friend Function:

- Private and protected members cannot be accessed from outside the class and it can be accessed only by the member function of same class
- But if a non- member function is defined as a friend function then, it can access private and protected data of class
- For accessing the private or protected data, the declaration of a friend function should be made inside the body of a class (no matter public or private or protected)

Characteristics of friend function:

1. A friend function can be declared in private or public or protected section of the class.
2. Since it is not a member function of class in which it is declared as friend, it is not called using object of class.
3. The friend function is called directly as normal function without the use of an object.
4. It is not a member function but can access the private and protected members of the class.
5. A friend function cannot access the class members directly. It needs to make use of class object and dot operator to access the class members.
6. It usually takes object as argument.

Degrades the importance of data hiding hence should be used only when necessary.

Friend Function:

We declare a friend function using friend keyword inside the body of the class.

Syntax for declaration of friend function:

```
class className
{
....
friend returnType  functionName(arguments);
…
};
```

Q1. Write a program to define a class Number that has two integer numbers as data members. Use friend function to find which one is greater among the two data members.

```
#include<conio.h>
#include<iostream.h>
class Number
{
        int num1,num2;
  public:
        void setNum();
         void displayNum();
    friend void maximum(Number N);
        //friend function declaration
};
```

```
void Number:: setNum()
{
        cout<<"Enter two integer
        numbers :";
         cin>>num1>>num2;
}
void Number:: displayNum()
{
        cout<<endl<<"The numbers
are”<<num1 <<" and "<<num2<<endl;
}
```

Q1.continued

```
void maximum(Number N)
{
if(N.num1>N.num2)
        cout<<endl<<N.num1<<" is greater than"<<N.num2<<endl;
 else
        cout<<endl<<N.num2<<" is greater than "<<N.num1<<endl;
}
```

```
void main()
{
        Number obj;
        obj.setNum();
        obj.displayNum();
        maximum(obj);
getch();
}
```

Example:

Q2. Write a program to define a class Time with data members hour, minute and second. Define a friend function to add two times.

Q3. Write a program to add two complex numbers using friend function.

Q4. Write a program to compare data of two different classes using friend function.

```
#include<conio.h>
#include<iostream.h>

class One; //forward declaration
class Two
{
        private:
                int num2;
        public:
                void setNum();
                void displayNum();
    friend void maximum(One a, Two b);
};
```

```
class One
{
        private:
                int num1;
  public:
                void setNum();
                void displayNum();
     friend void maximum(One a, Two b);
};
```

Q4. Program continued…

```
void One:: setNum()
{
        cout<<"Enter a number for class One";
         cin>>num1;
}
void One:: displayNum()
{
        cout<<endl<<"The data value of num in class One is "<<num1<<endl;
}
```

Q4. Program continued…

```
void Two:: setNum()
{
        cout<<"Enter a number for class Two";
         cin>>num2;
}
void Two:: displayNum()
{
        cout<<endl<<"The data value of num in class Two is "<<num2<<endl;
}
```

Q4. Program continued…

```
void maximum(One X, Two Y)
{
  if(X.num1>Y.num2)
                cout<<X.num1<<" is greater than
"<<Y.num2;
   else
                cout<<Y.num2<<" is greater than
"<<X.num1;
}
```

```
void main()
{
   One a;
   Two b;
   a.setNum();
   cout<<endl;
   b.setNum();
   cout<<endl;
   a.displayNum();
   b.displayNum();
   cout<<endl;
   maximum(a,b);
   getch();
}
```

Practice:

Q5. Write a program to swap the private data of two classes using friend function.

Q6. Write a program which has two classes named Practical and Theory which have data members to store practical and theory marks respectively. Use a non member function to calculate total marks (i.e practical marks of Practical class and theory marks of Theory class)

Friend Function:

Member function of one class can be friend function of another class. In this case, friend function is declared using scope resolution operator.

Syntax:

```
class One;
class Two
{
        …
        int functionTwo(One obj); //member function of class Two
};
class One
{
        …
        friend int Two::functionTwo(One obj); //functionOne() of class One is friend function of class Two
};
```

Friend Function:

```
class One;
class Two
{
        …
        int functionTwo(One obj); //member function
//of class Two
};
class One
{
        …
        friend int   Two::functionTwo(One obj);
//functionOne() of class One is friend function of class Two
};
```

Here, functionTwo() is member function of class Two and it is friend function of class One. So, it can access private and protected members of class One.

Example:

Q7. Write a program to swap contents of variables of two different classes where swapping function is member function of one class and friend function of another class.

Q8. Write a program which has two classes named Practical and Theory, which have data members to store practical and theory marks respectively. Define a class Marks that has member function to calculate total marks (i.e practical marks of Practical class plus theory marks of Theory class). Use the concept “member function of one class is friend of another class”.

Q7. Program:

```
#include<iostream.h>
#include<conio.h>

class One; //forward declaration
class Two
{
        private:
                int num2;
         public:
                void setNum();
                void displayNum();
                 void swap(One &a);
};
class One
{
        private:
                int num1;
          public:
                void setNum();
                 void displayNum();
                 friend void Two :: swap(One &a);
};
```

Q7. Program continued

```
void One:: setNum()
{
                cout<<"Enter a number for class One :";
                 cin>>num1;
}
void One:: displayNum()
{
                cout<<endl<<"The  data value of num in class One is : "<<num1<<endl;
}
void Two:: setNum()
{
                cout<<"Enter a number for class Two :";
                 cin>>num2;
}
void Two:: displayNum()
{
                cout<<endl<<"The  data value of num in class Two is : "<<num2<<endl;
}
void Two ::swap(One &X)
{
                int temp;
                temp=X.num1;
                X.num1=num2;
                num2=temp;
}
```

Q7. Program continued

```
void main()
{
  One a;
  Two b;
  a.setNum();
  cout<<endl;
  b.setNum();
  cout<<endl<<endl;
  cout<<"Before swapping :"<<endl;
  a.displayNum();
  b.displayNum();
  cout<<endl;
  b.swap(a);
  cout<<endl;
  cout<<"After swapping :"<<endl;
  a.displayNum();
  b.displayNum();

  getch();
}
```

Friend Class:

- A class can also be declared as friend of another class.
- A **friend class** is a **class** that can access the private and protected members of a **class** in which it is declared as **friend**.
- Friendship is not mutual. If class A is a friend of B, then B doesn’t become a friend of A automatically.
- When a class **XYZ** declares another class **ABC** as its friend, then friend class **ABC** can access private and protected members of the class **XYZ**.

Friend Class:

- Syntax:

```
class ABC;
class XYZ
{
        ...
        friend class ABC;
};
class ABC
{
        ...
};
```

*When a class **XYZ** declares another class **ABC** as its friend, then friend class **ABC** can access every members of the class **XYZ** including private and protected members.*

Example Program:

```
#include<iostream.h>
#include<conio.h>

class ABC;
class XYZ
{
                int a,b;
                public:
                void setData()
                {
                cout<<"Enter two numbers :";
                cin>>a>>b;
                }
    friend class ABC;
};
```

```
class ABC
{
                public:
                void display(XYZ x)
                {
                x.setData();
                cout<<endl<<"The first number is "<<x.a<<endl;
                cout<<"The second number is "<<x.b<<endl;
                }
};
void main()
{
                ABC obj1;
                XYZ obj2;
                obj1.display(obj2);
                 getch();
}
```

Friend Class:

Q1. Write a program which has three classes named Practical and Theory and Marks, which have data members to store practical marks, theory marks and total marks respectively. Calculate total marks (i.e practical marks of Practical class plus theory marks of Theory class). Use the concept of friend class.

Friend Function Practice:

Q1. Create classes called Class1 and Class2 with each having one private member. Add member function to set a value (say setValue) on each class. Add one more function max() that is friendly to both classes, max() function should compare two private member of two classes and show maximum among them. Create one-one object of each class and set a value on them. Display the maximum number among them.

Friend Function Practice:

Q2. Create a class called Mountain with data members name, height and location. Define a member function to initialize the data members. A friend function cmpHeight() to compare height of two objects and member function displayInfo() to display information of mountain. In main create two objects of the class mountain and display the information of mountain with greatest height.

End of Chapter 2