Chapter - 9

Classes and Objects

9.1 Introduction

Class is the most important feature of object-oriented programming language. The concept of class is taken from the structure in C. It is a new way of creating and implementing user-defined type.
In this chapter, we shall discuss the various concepts of classes and objects.

9.2 Defining class

A class is a user-defined data type that binds data and function together. The class declaration includes the declaration of its data members and member functions.
The syntax of the class declaration is :

```
class class_name
{
  private:
          variable declaration;
          function declaration;
public:
           variable declaration;
          function declaration;
};
```

The class members that are declared in private section can be accessed only by the members of that class. The class members that are declared in public section can be accessed from outside the class also. By default, the members of a class are private. The data hiding by using private declaration is the important feature of object-oriented programming.

A simple class example

```
class point
{
        int x,y;                    // private by default
public:
        void input(int a, int b);
        void output(void);
};
```

Creating objects

Like basic data type, we can create the variables of class type. These variables are called objects.
For example

```
point p, q;
```

In the above statement, two objects p and q of class type point are created.

Accessing class members

The public members of the class can be accessed from outside the class by using the object of that class.

The syntax for accessing a public member function is:

```
object_name.function_name(arguments list);
```

For example, the function call statement

```
p.input(10,20);
```

assign the value 10 to x and 20 to y of the object p by defining the input() function.

The statement p.x=10; is illegal, since x is declared private and it can only accessed by the member functions directly and not by the object from outside the class.

9.3 Defining member functions

The member function of a class can be defined within the class and outside the class also.

Inside the class

The function declaration is replaced by the actual definition of the member function inside the class. The function defined inside the class are treated as an inline function.

For example

```
class point
{
        int x,y;
public:
        void input(int a, int b)
        {
                x=a;
                y=b;
        }
        void output(void)
        {
                cout<<"x="<<x<<"\n";
                cout<<"y="<<y;
        }
};
```

Outside the class

The member functions declared in a class must be defined separately outside the class. The format for member function definition is:

```
return_type class_name:: function_name(arguments)
{
        function body
}
```

The class name indicates the function belongs to this particular class.
For example

```
class point
{
        int x,y;
public:
        void input(int a, int b);
        void output(void);
};
void point :: input(int a, int b)
        {
                x=a;
                y=b;
        }
        void point :: output(void)
{
        cout<<"x="<<x<<"\n";
        cout<<"y="<<y;
}
```

Program 9.1 A simple program with class

```
#include<iostream>
using namespace std;
class point
{
        int x,y;
public:
        void input(int a, int b);
        void output(void);
};
void point :: input(int a, int b)
        {
                x=a;
                y=b;
        }
        void point :: output(void)
        {
                cout<<"x="<<x<<"\n";
```

```
                    cout<<"y="<<y;
          }
int main()
{
          point p;
          p.input(5,10);
          p.output();
return 0;
}
```

The output of the program 9.1 would be:

```
x=5
y=10
```

9.4 Access Modifiers

public and private keywords are called access modifiers. Since, they control the access mechanism of members of a class.

- The public member of a class can be accessible from outside the class. Generally, member function of a class are kept in public section of the class.
- The private members of a class can't be accessible from outside the class even with the object of that class. Generally, variables are kept in private section of the class.

9.5 Arrays within class

Arrays can act as a data member of a class.

Program 9.2 Array within class

```
#include<iostream>
using namespace std;
class data
{
          int a[5];
public:
          void getdata(void);
          void showdata(void);
};
void data :: getdata(void)
{
        cout<<"Enter the elements of array\n";
        for(int i=0; i<5; i++)
        {
         cin>>a[i];
        }
}
void data :: showdata(void)
{
        cout<<"Array elements are\n";
        for(int i=0; i<5; i++)
        cout<<a[i]<<"\t";
}
int main()
{
    data d;
    d.getdata();
    d.showdata();
return 0;
}
```

The output of the program 9.2 would be:

```
Enter the elements of array
6       5       9       8       1
Array elements are
6       5       9       8       1
```

9.6 Static Data Members

The data members of a class can be declared as a static. The characteristics of static data members are:

- Its initial value is set to zero, when first object of its class is created.
- Only single copy of the data member is created and it is shared by all the objects of the class.
- Since it is associated with the entire class, it is also called class

variables.

Program 9.3 Static Data Members

```
#include<iostream>
using namespace std;
class data
{
	static int x;
	int y;
	public:
	void getdata(int a)
	{
		y=a;
		x++;
	}
	void show_x(void)
	{
		cout<<“x=”<<x<<“\n”;
	}
};
int data :: x;	// static member definition
int main()
{
	data d1, d2;	// x is initialized to zero
	d1.show_x();
	d2.show_x();
	d1.getdata(10);
	d2.getdata(20);
	cout<<“After reading data”<<“\n”;
	d1.show_x();
	d2.show_x();
	return 0;
}
```

The output of the program 9.3 would be:

```
x=0
x=0
After reading data
x=2
x=2
```

The static data member x is initialized to zero when objects are created. The value of x is incremented by one each time the function getdata() is called.

Since the static variable x is shared between the two objects, the value of x is printed 2 each time. The initial value can be assigned to static data member, when it defined outside the class. For example:

int data :: x=5;

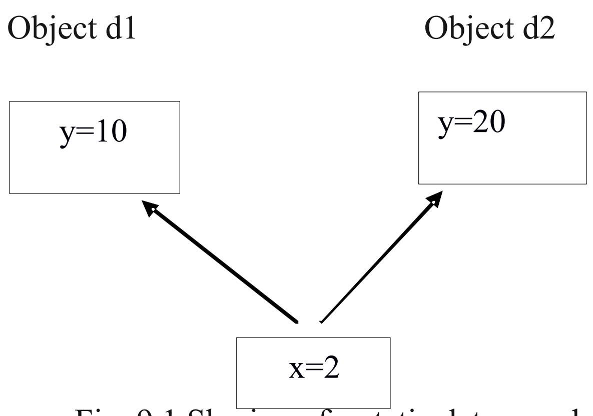

Fig. 9.1 Sharing of a static data member

9.7 Static Member Function

A member function declared with static is called static member function. The properties of static member functions are:

- It can access only other static data members and member functions in the same class.
- They are invoked using the class name.

Program 9.4: Static Member Function

```
#include<iostream>
using namespace std;
class test
{
        int x;
        static int y;
public:
        void set_xy(int a)
        {
                x=a;
                y++;
        }
        void show_x(void)
        {
                cout<<"x="<<x<<"\n";
```

```
          }
          staic void show_y(void)
          {
                    cout<<"y="<<y;
          }
};
int test :: y;
int main()
{
test t1, t2;
t1.set_xy(10);
t2.set_xy(20);
t1.show_x();
t2.show_x();
test::show_y();          // calling static function
return 0;
}
```

The output of the program 9.4 would be:

```
x=10
x=20
y=2
```

9.8 Friend function

As we know that the private members of a class can't be accessed from outside the class. A friend function can access the private data of a class through the object of that class. A function that is common to two classes , we can make this as a friend to these two classes. The function is declared with keyword friend. A friend function has following characteristics:

- It is invoked like a normal function, not with the any object of the class.
- It can only access the members of the class by using the object of that class.
- It can be declared anywhere in the class.
- Generally, it has objects as arguments.

Program 9.5: Friend function

```
#include<iostream>
using namespace std;
class test
{
          int  x,y;
          public:
          void getdata(int a, int b)
```

```
        {
                x=a;
                y=b;
        }
        friend int sum(test t);
};
        int sum(test t)
        {
                return(t.x+t.y);
        }
        int main()
        {
                test q;
                q.getdata(10,20);
                cout<<"Sum="<<sum(q);
                return 0;
        }
```

The output of the program 9.5 would be:

```
Sum=30
```

Friend class

A member function of a class can be friend function of another class.

For example

```
class A
{
void fun();     // member function of A
};
class B
{
- ------
friend void A::fun();
--------
};
```

Note that the friend function in class B is declared with the class name and scope resolution operator.

The function fun() is a member function of class A and friend function of class B.

If all the member function of one class are declared as friend functions in another class, then the class is called friend class. For example

```
class C
{
```

```
- ------
friend class A;                    //All member functions of class A are friend to C
- ------
};
```

Program 9.6: Using friend function to find maximum between data members of two classes

```
#include<iostream>
using namespace std;
class second; //forward declaration
class first
{
        int x;
public:
        void set_value(int a)
        {
                x=a;
        }
        friend void max(first, second);
};
class second
{
        int y;
public:
        void set_value(int b)
        {
                y=b;
        }
        friend void max(first, second);
};
void max(first f, second s)
{
        if(f.x>s.y)
        cout<<"Maximum is "<<f.x;
        else
        cout<<"Maximum is "<<s.y;
}
int main()
{
        first A;
        second B;
        A.set_value(10);
        B.set_value(20);
        max(A,B);    // calling friend function
        return 0;
}
```

The output of the program 9.6 would be:
Maximum is 20

9.9 Returning Objects

In the previous section, we saw that the friend functions receive objects as arguments. A friend function can return object also.

Program 9.7: Returning object

```
#include<iostream>
using namespace std;
class vector
{
        int V[3];
public:
        void set_vector(void)
        {
                cout<<"Enter three numbers\n";
                for(int i=0; i<3; i++)
                cin>>V[i];
        }
        void display(void)
        {
                for(int i=0; i<3; i++)
                cout<<V[i]<<",";
        }
        friend vector sum(vector, vector);
};
vector sum(vector p, vector q)
{
        vector r;
        for(int j=0;j<3; j++)
        r.V[j]=p.V[j]+q.V[j];
        return r;
}
int main()
{
        vector v1, v2,v3;
        v1.set_vector();
        v2.set_vector();
        v3=sum(v1,v2);
        cout<<"First vector is:";
        v1.display();
        cout<<"\n";
        cout<<"Second vector is:";
        v2.display();
        cout<<"\n";
        cout<<"Resultant vector is:";
        v3.display();
        return 0;
}
```

The output of the program 9.7 would be:

```
Enter three numbers
3       -2      5
Enter three numbers
-8      6       7
First vector is: 3,-2,5,
Second vector is: -8,6,7,
Resultant vector is: -5,4,12
```

9.10 Pointers to members

We can assign the address of a class member to a pointer
For example

```
class X
{
        int a;
        public void show();
};
```

We can define pointer to member as:

```
                                int X:: *p=&X::a;
```

X::* means "pointer-to-member of X".
&X::a means "address of the member a of the class X"
The statement int *p=&a; will not work;
The pointer p can be used to access the member a inside member function or friend function.
For example

```
void show()
{
        X x;            // object created
        cout<<x.*p;   //display value of a
        cout<<x.a;    //same as above
}
```

We can also set a pointer to member function of a class. The member function can be invoked using dereferencing operator (.*).

For example

```
X x;            // object created
void (X::*pf)()=&X::show;
(x.*pf)();      //invoke show()
```

Here, pf is a pointer to member function show().

Important Points

- A class is a user-defined data type that binds data and function together.
- By default, the members of a class are private.
- The member function of a class can be defined within the class and outside the class also.
- public and private keywords are called access modifiers.
- The data members of a class can be declared as a static.
- A member function declared with static is called static member function.
- A friend function can access the private data of a class through the object of that class.
- We can assign the address of a class member to a pointer.

Practice Questions

Objective type questions:

Q.1 User-defined data type that binds data and function together is called

A. Object	B. Class
C. Array	D. Pointer

Q.2 By default, the members of a class are

A. public	B. private
C. protected	D. None of these

Q.3 Which is an Access Modifier?

A. public	B. private
C. Both A and B	D. None of these

Q.4 Which is true with respect to static data members?

A. Its initial value is set to zero, when first object of its class is created

B. Only single copy of the data member is created.

C. Also called class variables.

D. All of these

Q.5 Which is true with respect to static member functions?

A. Declared with static keyword

B. Access only other static data members and member functions in the class.

C. Invoked using the class name.

D. All of these

Q.6 Which is true with respect to friend functions?

A. Invoked like a normal function.

B. Declared anywhere in the class.

C. Generally, it has objects as arguments.

D. All of these

Very Short Answer Type Questions

Q.1 What is class?

Q.2 What is object?

Q.3 What is friend class?

Short Answer Type Questions

Q.1 Differentiate between private and public access modifiers.

Q.2 What are the characteristics of static data members?

Q.3 What are the properties of static member functions?

Essay Type Questions

Q.1 What is friend function? Write its characteristics.

Q.2 Write a program to create a class 'complex', that represents a complex number and define the member functions to compute addition and subtraction of two complex numbers.

Q.3 Write a program to swap data members of two classes using friend function.

Answer Key

1. B	2. B	3. C
4. D	5. D	6. D