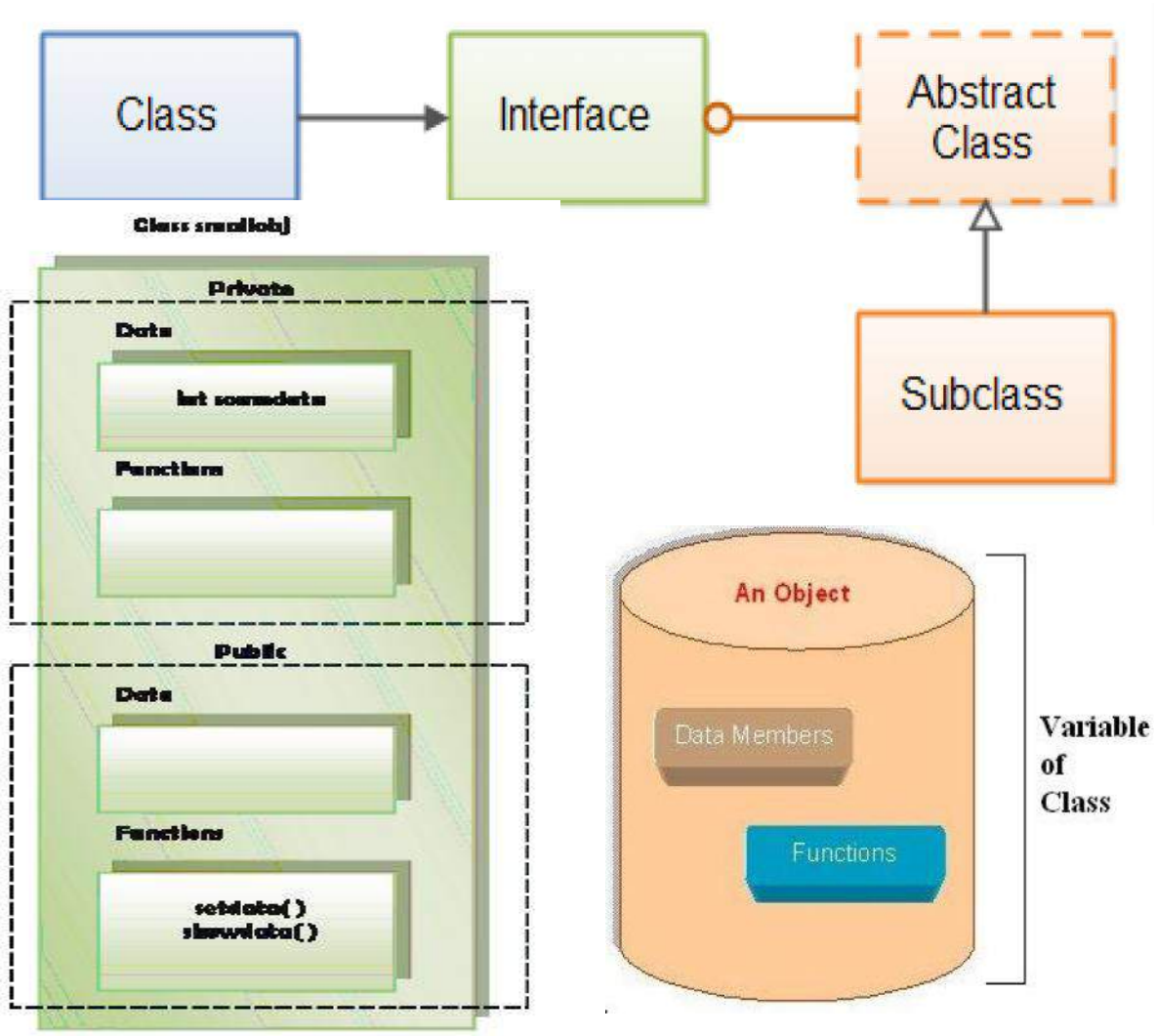
Chapter 7
Classes and objects

Objectives:

- **To understand the need of classes.**
- **To understand the definition and declaration of classes**
- **To understand the use of members functions inside and outside classes**
- **To understand the process of accesing and manipulating the members of class**
- **To illustrate use of classes through application of simple programs.**



7.1 Introduction

In the previous chapter we have learnt the fundamentals units of an object oriented programming. In this chapter let us understand the basic units of data representation for development of any object oriented programming namely classes and objects.

Real world objects can be represented as data variables in computer domain. Data variables are identified as account numbers, balance_payment, Employee_code and so on. When these real world objects are represented as data in computers, they can also be manipulated or processed by using functions in C++. For instance account information of a customer in a bank can be updated, balance_payment can be computed and so on. To combine the data variables (objects) along with appropriate functions for manipulation of these objects, the concept of classes were introduced in object oriented programming.

The elementary concept of object oriented programming begins with the use of classes and objects. A class is a very powerful keyword in C++. A class declaration defines new user-defined data types that link data and the code that manipulates the data. In other words, a class combines data elements and functions (operations) into a single entity.

We have already discussed the concept of combining of data elements into a group called structures in the previous year. Application of user-defined functions for performing different operations has also been discussed. Classes combine both data elements and operations associated with the data.

Let us look at the terminologies in procedural programming language and object oriented programming.

Procedural programming	**Object oriented programming**
Variables	Objects
User-defined data types	Classes
Structure members	Instance variables
Functions	Methods
Function call	Message passing

The data elements in a class are called member data and the functions in a class are called member functions. This chapter explains classes, objects, member functions, data hiding, abstraction, encapsulation and how these features are implemented using classes (further chapters provide details of single inheritance, multiple inheritances, insight to polymorphism and other functions).

7.2 Definition and declaration of classes and objects:

A class definition is a process of naming a class and data variables, and interface operations of the class.

A class declaration specifies the representation of objects of the class and set of operations that can be applied to such objects.

The definition and declaration of a class indicates the following:

- The data variables known as member data of a class, describe the characteristics of a class.
- Member functions are the set of operations that are performed on the objects of the class. There may be zero or more member functions in a class. This is also known as interface.
- The access control specifiers to the class members within a program are specified.
- Class tagname for external operations for accessing and manipulating the instance of a class.

The General syntax for defining a class is as follows:

```
class user_defined_name
{
        private:
                Member data
                Member functions
        protected:
                Member data
                Member functions
        public:
                Member data
                Member functions
};
```

Keyword **class** is used to declare a class. User_defined_name is the name of the class.

Class body is enclosed in a pair of flower brackets. Class body contains the declaration of its members (data and functions). There are generally three types of members namely private, public and protected. These are discussed in the following section.

The class function definition describes how the class functions are implemented.

Example 7.1: Let us declare a class for representation of bank account.

```
class account
{
        private:                        //implicit by default
                int     accno;
                char    name[20[;
                char    acctype[4];
                int     bal _amt;
        public:                                 //member functions
                void get_data();
                void displaydata();
}
```

In the above example, class name account is a new type identifier that is used to declare instance of that class type. The class account contains four member data and two methods or member functions. Both of these member data are private by default, while both member functions are public by default.

The functions get_data( ) and put_data( ) are used to write instructions to perform operations on member data. These two functions only can provide access to get member data from outside the class. Therefore, the data cannot be accessed by any other function that is not a member of the class account.

Program 7.1 Program to show the use of class and object

```
#include<iostream.h>
#include<conio.h>
class Test
{
        private:
                int a,b;
        public:
                void getnum()
                {
                        cout<<"Enter Numbers for Addition: ";
                        cin>>a>>b;
                }
                void show_data()
                {
                        cout<<"Addition of two numbers: "<<(a+b);
                }
};
```

```
int main()
{
        clrscr();
        Test a1;
        a1.getnum();
        a1.show_data();
        getch();
        return 0;
}
```

Enter Numbers for Addition: 50 60

Addition of two numbers: 110

In the above program a, b are data members of the class Test. get_num() and show_add() are member functions. get_num() is used to input data values. show_add() function is used to add two numbers and display the sum of two numbers.

7 .3 Access specifiers

An object is defined to be an instance of a class. Every data member of a class is specified by three levels of access protection for hiding data and function members internal to the class. The access specifiers define the scope of data. The access specifers are indicated using the following keywords.

7.3.1 private

private access means a member data can only be accessed by the member function. Members declared under private are accessible only within the class. If no access specifier is mentioned, then by default, members are private.

Example 7.2:

```
private:
        int     x;
        float   y;
```

7.3.2 public

public access means that members can be accessed by any function outside the class also.

Example 7.3:

```
class  box
{
        int     length;
        public : int  width;
        private : int height;
        void set_height ( int  i )
        {
                height = i ;
        }
```

```
        int get_height ( )
        {
                return height ;
        }
};
int  main()
{
        box    object;
        object.length = 10;
        object.width = 20;
        object.set _height ( 30);  // private variable can be accessed
        return 0;                        only  through its public method
}
```

Thus keyword public identifies both class data and functions that constitute the public interface for the class. In the above program data member width is a public variable. set_height() and get_height() are member functions. The private variable height can be accessed only through the member functions of the class.

Thus keyword public identifies both class data and functions that constitute the public interface for the class.

7.3.3 Protected

The members which are declared using protected can be accessed only by the member functions, friends of the class and also by the member functions derived from this class. The members cannot be accessed from outside. The protected access specifier is therefore similar to private specifier. The difference is discussed in detail in further chapters.

7.4. Members of the class

The members of a class can be data or functions. Private and protected members of a class can be accessed only through the member functions of that class. No function outside a class can include statements to access data directly. The public data members of objects of a class can be accessed using direct member access operator (.)

The syntax for accessing class member is

```
class-object. member-data
class-object.member-function(arguments);
```

Example 7.4:

```
class rectangle
{
        int     l;
        int     b;
    public:
```

```
                    void   get_data();
                    void   compute_area();
                    void  display();
        };
int  main()
{
        rectangle    rl;
        rl.get _data();
        rl.compute_data();
        r1.display();
        return 0;
}
```

In the above example l and b are data members of class rectangle and get_data(), compute_data() and display() are member functions. r1 is a class variable that invokes the member functions.

Program 7.2. Program to show the use of access specifiers with classes and objects

```
#include<iostream.h>
#include<conio.h>
class data
{
        private:
                int day;
                int month;
                int year;
        public:
                void date(int dd,int mm,int yy)
                {
                        day = dd;
                        month = mm;
                        year = yy;
                        cout<<"\t"<<day<<"\t"<<month<<"\t"<<year<<endl;
                }
};
int  main()
{
        clrscr();
        data date1,date2;
        date1.date(7,12,2014);
        date2.date(8,12,2014);
        date1.date(12,12,2014);
        getch();
```

```
        return 0;
}
```

In the above program day, month and year are private members and date is a public function. The output of the above program is as follows.

```
7    12    2014
8    12    2014
12   12    2014
```

7.5 Member functions

Member functions can be defined in two places:

- Inside class definition
- Outside class definition

7.5.1 Inside class definition

To define member function inside a class the function declaration within the class is replaced by actual function definition inside the class. A function defined in a class is treated as inline function. Only small functions are defined inside class definition.

Syntax: return_type classname(member function)

Example 7.5:

```
class rectangle
{
        int    length;
        int    breadth;
    public:
        void get_data ()
        {
            cin>>length;
            cin>>breadth;
        }
        void put_data (void)
        {
            cout <<length ;
            cout <<breadth;
        }
};
```

7.5.2 Outside class definition

A function declared as a member of a class is known as member function. Member functions declared within a class must be defined separately outside the class. The definition of member function is similar to normal function. But a

member function has an 'identity label' in the header. This label tells the compiler which class the function belongs to. The scope of the member function is limited to the class mentioned in the header. Scope resolution operator :: is used to define the member function.

Syntax: For member function outside a class

```
return_type classname :: memberfunction(arg1, arg2, ..., argn)
{
        function body;
}
```

The member functions have following characteristics:

- Type and number of arguments in member function must be same as types and number of data declared in class definition.
- The symbol :: is known as scope resolution operator. Usage of :: along with class name is the header of function definition. The scope resolution operator identifies the function as a member of particular class.
- Several classes can use same function name. Membership label will resolve their scope.
- Member function can access private data of a class. A non-member function cannot.

Example 7.6: The following program segment shows how a member function is defined outside class.

```
class operation
{
        private :
                int a;
               int b;
         public :
                int sum();
                int product();
};
int operation::sum()
{
        return (a+b) ;
}

int operation (product)
{
        return (a*b) ;
}
```

In the above example sum() and product() are member functions defined outside class operation. **The use of scope resolution operator implies that these member functions are defined outside the class.**

Program 7.3 To use classes using member functions inside and outside class definition.

```
# include<iostream.h>
class item   // class declaration
{
        int number;              // private by default
        float cost;
    public :
        void getdata(int a,  float b);
        void putdata(void)       // function defined here
        {
            cout <<"Number: "<< number << endl;
            cout << "Cost: "<< cost << endl;
        }
};
// member function outside class definition
void item :: getdata(int a, float b)
{
    number = a;
    cost = b;
}
// main program
int main()
{
    item x;                      // create object x
     x.getdata(250, 10.5);
    x.putdata();
    return 0;
}
```

In the above program one member functions putdata() is defined inside class definition and the other member function getdata() is defined outside class definition. Here is the output of program.

```
Number : 250
Cost : 10.5
```

7.6. Defining objects of a class

When a class is defined, it specifies the type information the objects of the class store. Once the class is defined an object is created from that class. The

objects of a class are declared in the same manner like any other variable declaration.

Syntax :
```
class user_defined_name
{
        private:                    //Members
        public:                     // Methods
};
User_defined_name object1, object2,  ... ;
```

Example 7.6:
```
class num
{
        private:
                int x;
                int y;
         public:
                 int sum(int p, int q);
                 int diff(int p, int q);
};
void main()
{
        num s1,s2;
        s1.sum(200,300);
        s2.diff(600,500);
}
```

Note that an object is an instance of a class template.

Example 7.6 A class to create studentname, rollno, sex, age.

```
class student
{
        int rollno;
        char name[20];
        char gender;
        int age;
    public:
        void get_data();
        void display_data();
};
student obj1, obj2;        //Obj1, obj2 are objects of class student
```

7.7 Arrays as members of classes

So far we have considered primary data values as member of class. Just like arrays within structures, it is possible to use arrays as member data of class type.

Example 7.7:

```
class marks
{
        int m[5];
        int i  ;
    public:
        void setval(void);
        void display(void);
};
```

The array variable m is a private member of class marks. This can be used by member function setval( ) and display( ) as follows.

```
void marks :: setval(void)
{
    cout<< "Enter marks: "<<endl;
    for (i=0;i<5;i++)
        cin>>m[i] ;
}
void marks::display(void)
{
    cout<< "The marks are : ";
    for(i =0;i<5;i++)
        cout<<setw(4)<<m[i]<<endl;
}
```

7.8 Array of objects

An array having class type elements is known as array of objects. An array of objects is declared after class definition and is defined in the same way as any other array.

Example 7.8:

```
class employee
{
        char name[10] ;
        int age;
    public:
        void getdata();
        void dispdata(void);
};
employee supervisor[3] ;
employee sales_executive[5] ;
employee team_leader[10] ;
```

In the above example, the array supervisor contains 3 objects namely supervisor[0], supervisor[1],supervisor[2].

	Name	Age
supervisor[0]		
supervisor[1]		
supervisor[2]		

Storage of data items in an object array

Program 7.4 Program to show the use of array of objects

```
#include<iostream.h>
#include<conio.h>
class data
{
                int  rollno, maths, science;
        public:
                int avg();
                void getdata();
                void putdata();
};
void data::getdata()
{
         cout<<"Enter rollno: ";
        cin>>rollno;
        cout<<"Enter maths marks: ";
        cin>>maths;
        cout<<"Enter science marks: ";
        cin>>science;
        putdata();
}
int data::avg()
{
        int a;
        a=(maths+science)/2;
        return a;
}
void data::putdata()
{
        cout<<"Average = "<<avg()<<endl;
}

int main()
{
```

```
        clrscr();
        data stud[3];
        for(int  i=0;i<3;i++)
                stud[i].getdata();
        return 0;
}
```

```
Enter rollno: 23
Enter maths marks: 56
Enter science marks: 78
Average = 67
Enter rollno: 24
Enter maths marks: 56
Enter science marks: 77
Average = 66
Enter rollno: 12
Enter maths marks: 44
Enter science marks: 89
Average = 66
```

In the above program stud is an array of objects, data is a class with member functions getdata() and putdata() defined outside the class definition.

7.9. Objects as function arguments

A function can receive an object as a function argument. This is similar to any other data type being sent as function argument. An object can be passed to a function in two ways:

- Copy of entire object is passed to function (pass-by-value).
- Only address of the object is transferred to the function (pass-by-reference).

In pass by value, a copy of the object is passed to the function. The function creates its own copy of the object and uses it. Therefore changes made to the object inside the function do not affect the original object.

In pass by reference, when an address of an object is passed to the function, the function directly works on the original object used in function call. This means changes made to the object inside the function will reflect in the original object, because the function is making changes in the original object itself.

Pass-by-reference is more efficient, since it require only to pass the address of the object and not the entire object.

Program 7.4: Program to show objects as arguments to function.

```
#include<iostream.h>
#include<conio.h>
class currency
 {
                int rupee, paise;
                int total;
        public:
                void get_value(int r, int p);
                void display(void);
};
void currency::get_value(int r, int p)
{
        rupee = r;
        paise = p;
        total = r*100 + p;
}
void currency::display(void)
{
        cout<<rupee<<" Rupees "<<" and "<<paise<<" paise"<<endl;
        cout<<"Converted value: "<<total<<" paise";
}
int main()
{
        currency c;
        c.get_value(5, 75);
        c.display();
}
```

```
5 Rupees  and 75 paise
Converted value: 575 paise
```

Program 7.5: Program to show objects as arguments to function.

```
#include <iostream.h>
#include<conio.h>
class rup
{
        private:
           int n1,n2;
    public:
          rup():n1(1),n2(1)
```

```
        {
        }
        void get()
        {
                cout<<"enter first number";
                cin>>n1;
                cout<<"enter second number";
                cin>>n2;
        }
        void print()
        {
                cout<<"product="<<n1<<endl;
                cout<<"product2="<<n2<<endl;
        }
        void multi(rup r1,rup r2)
        {
                n1=r1.n1*r1.n2;
                n2=r2.n1*r2.n2;
        }
};
int main ()
{
        rup r1,r2,r3;
        clrscr();
        r1.get();
        r2.get();
        r3.multi(r1,r2);
        r3.print();
        getch();
        return 0;

}
```

7.10. Differences between structure and class:

In C++, a structure is a class defined with the keyword struct. Its members and base classes are public by default. A class is defined with the class keyword

Points to remember:

- A class is a user defined data that contains data members and objects
- An object is an instance of a class.
- Members: There are two types of members in a class. They are data member and member functions.
- Data members are different data variables similar to structure members.
- Data members may be declared in a class as private, public or protected.

- Private members are default by declaration. They are accessible only to member functions within a class only.
- Public members can be accessed inside as well as outside class definition.
- Member function also known as method, acts on data members in the class.
- Member functions may be defined inside or outside a class.
- Scope resolution operator :: is used when a global variable exists with the same name as local variable. This is also used in C++ when member functions are declared outside the class definition. The function name is preceded by the class name and scope resolution operator.
- Class members are accessed using dot(.) operator.
- An array having class type elements is known as an array of objects.
- Functions can receive objects as function arguments.
- It is possible to pass objects to a function using pass by value or pass by reference.

One mark questions:

1. What is a class?
2. What is an object?
3. What are the two types of members referenced in a class?
4. What are data members?
5. What is a member function?
6. Mention the access specifiers used with a class
7. Is it possible to access data outside a class?
8. Which type of data members are accessible outside a class?
9. Which access specifier is implicitly used in a class?
10. Define the term public access
11. Mention the operator used to access members of a class
12. What is the significance of scope resolution operator (::)?
13. How are objects of a class declared? Give an example
14. What is meant by an array of objects?
15. Write an example to show how objects can be used as function arguments?

Two/Three mark questions

1. Write the differences between class definition and class declaration.
2. Write the syntax and example for class definition.
3. Write the syntax and example for class declaration.
4. What is the significance of using access specifiers? Mention different access specifiers.
5. Discuss private access specifier with an example.

6. Write short notes on public access specifier.
7. Explain protected access specifer.
8. How are class members referenced? Discuss with suitable example.
9. What is meant by referencing member functions inside class definition and outside class definition?
10. Discuss how objects of a class are referenced with an example.
11. How can arrays be used as class members. Write an example.
12. How are objects passed as arguments to a function? Give an example.

Five mark questions

1. Explain class definition and class declaration with syntax and example.
2. Describe access specifiers in a class.
3. Explain member functions

a. Inside class definition

b. Outside class definition

4. What are the characteristics of member functions outside a class?
5. Explain how objects of a class can be defined?
6. Illustrate how an array of objects can be defined.
7. Describe how objects can be used as function arguments.
8. Let product list be a linear array of size N where each element of the array contains following fields : Itemcode, price and quantity. Declare a class product list with the three data members and member functions to perform the following :
    a. Add values to the product list.
    b. Printing the total stock value.
9. A class clock has following members : a. hour b. minute
Create member functions
    a. To initialize the data members
    b. Display the time
    c. To convert hours and minutes to minutes
10. Write a program that receives arrival time, departure time and speed of an automobile in kilometers/hour as input to a class. Compute the distance travelled in meters/second and display the result using member functions.

****