3...

Classes and Objects

Objectives...

- To understand Concept of Classes and Objects.
- To understand about Access Specifiers.
- To study Data Member and Member Functions.
- To learn Static Data Member and Static Member Function.
- To learn Friend Function and Friend Class.

3.1 INTRODUCTION

- The classes and objects are the most important concepts in C++. Classes are basically the user defined data type which consists of data and member functions. Objects are instances of class, which holds the data variables declared in class and the functions work on these class objects.
- Classes and structures provide a convenient mechanism to the programmer to construct his/her own data types for convenience in representing real entities.
- A class serves as a blueprint or template that provides a layout common to all of its instances known as objects i.e., a class is only a logical abstraction that specifies what data and functions, its objects will have, whereas the objects are the physical entities through which those data and functions can be used in a program. Thus, objects are considered as the building blocks of object oriented programming.
- In this chapter we discusses the concept of classes and objects in detail.

3.2 STRUCTURE AND CLASS

- The key objective of OOP is to represent the various real-world objects as program elements. In C++, this objective is accomplished with the help of two user defined data types, structures and classes.
- C++ language structures and classes bind (combine) data and the functions together under a single entity and thus are considered as an extension of C structures, which cannot contain the functions.

- C++ language structures and classes are identical in terms of their functionality i.e., all the OOP concepts such as data abstraction, encapsulation, inheritance and polymorphism can be implemented in both. Thus, they can be used interchangeably with some modifications.
- However, to follow the conventions of OOP, classes are generally used to contain data and functions both, and structures are generally used to contain the data only.

1. Structure:

- A structure contains more than one data items called members, which are grouped together as a single unit.
- A structure in C++ operates much like a structure in C. The struct keyword is used for creating a structure in C++.
- C++ structure support data abstraction and procedural abstraction. Structure in C++ is define as,

```
struct employee
{
    private:
        char name[20];
        long phone;
    public:
        void display(void)
        {
            cout<<phone;
        }
};
```

- By default members of the C++ structures are public.

2. Class:

- A class is similar to a structure data type but consist of not only data elements but also functions which are operated on the data elements.
- A class is a logical method to organize data and functions in the same structure. The keyword 'class' is used for creating a class. Generally, class is used in C++ instead of struct.
- Class is user define data type with data elements and functions. All the members in a class are private by default.

For example,

```
class employee
{
    int EmpID;
    char name[20];   //By default both are private
    public:
        void getdata(void);
        void putdata();
};
```

Comparison between Structure and Class:

Table 3.1: Difference between structure and class

Sr. No.	Structure	Class
1.	The members of a structure have public access by default.	The members of a class have private access by default.
2.	'struct' keyword is used while declaring a structure.	'class' keyword is used while declaring a class.
3.	C++ structure is generally used when only attributes are associated with an entity and the values of these attributes can be accessed by any user of that structure.	C++ class is generally used when attributes and functions are associated with an entity and the values of one or more attributes should not be accessed by any user of that class.
4.	Public and private members declaration present in structure.	The data members and member functions can be defined as public, private as well as protected.
5.	Data hiding is not achieved.	Data hiding is achieved with private keyword.

3.3 CLASSES AND OBJECTS

- In this section we study classes and objects in detail.

3.3.1 Classes

Class Definition:

- Class is a user defined data type with data elements and functions (operations).
- A class is a way that binds data and functions that operate on the data together in a single unit. Like other user-defined data types, it also needs to be defined before using its objects in the program.
- A class definition specifies a new data type that can be treated as a built-in data type.
- The data members describe the state of the object. The operations define the behaviour of the object.
- Class is used to define abstractions. A class contains access specifiers, data members and member functions.

Declaration of Class:

- The keyword 'class' is used to declare a class in C++. The declaration of a class always ends with the semicolon (;). All data members and member functions are declared within the braces { and } which are called the body a class.
- General **syntax** for declaration of a class:

```
class class_name
{
   // body of a class
};
```

- The class specifies type and scope of its members. The keyword class indicates that the class name which follows in an abstract data type. The body of a class is enclosed within the curly braces followed by a semicolon (;). The body of a class contains declaration of variables and functions, collectively known as data members and functions are known as member functions. These members are grouped under three sections, private, protected and public which defines the visibility of members.
- Consider the following simple class example:

```
class student
{
    int rollno;
    char name[20];
    public:
    void getdata()
    {
        rollno = 10;
        name = "OMKAR";
    }
    void display()
    {
        cout<<"Roll no:"<<rollno;
        cout<<"Name:"<<name;
    }
}; //end of class
```

- In above example, a class called student contains two data members and two member functions.
- The data members are private by default whereas both the member functions are public.
- The general notation of representation of class student is shown in Fig. 3.1.

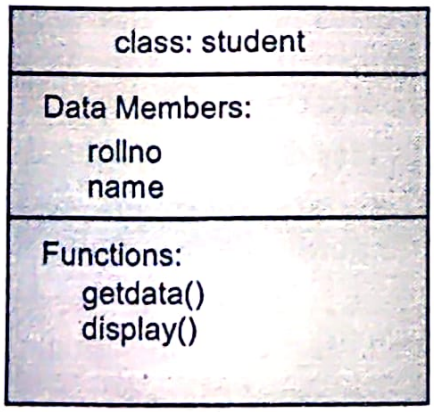

Fig. 3.1: Class

3.3.2 Objects

- An object is an instance of a class. A class provides blueprints for the object. An object is a variable of type class.
- When we create a object of a class:
  - Memory is allocated,
  - Constructor is called implicitly, and
  - Memory gets initialized.

- The process of creating object is called Class Instantiation. We can create object as:

```
class_name objectname; OR
class_name object1, object2 ..... objectn;
```

For example:

```
student S;
student S1, S2;      //more than one object of class student
```

- Another way is, we can create object by placing their name immediately after the class declaration.

```
class student
{
   ..........
   ..........
} S1, S2;
```

3.3.3 Class Members

- In the class declaration syntax, the variables and functions declared within the curly braces (class body) are collectively known as Members of the Class. The variables declared in the class are known as data members, while the functions declared in the class are known as member functions.
- In above example, we creates objects S1 and S2 of class student. Once, the object of a class is created, object can access the member of class using the member access operator (.) i.e., dot operator.
- **Syntax for accessing members through object:**

1. **To access data number we use following syntax:**

```
object_name.variable;
```

2. **To access member function we use:**

```
object_name.function_name(actual argument list)
```

- For example,

(i) If emp is object of class employee then we can write,

```
employee emp;
emp.eno; //eno should be public data member
```

(ii)
```
student S;
S.getdata();
```

(iii)
```
class employee
{
   int eno;
   char ename[10];
```

```
    public:
            int x;
            ........
};
.
.
.
employee e;
e.eno = 100;  //error since eno is private
e.x = 50;     // o.k. since x is public.
```

3.4 ACCESS SPECIFIERS [S-22; W-22]

- Access specifiers are used to restrict (limits) the accessibility of class members. Access specifiers define how the members of the class can be accessed.
- Each member of a class is associated with access specifiers. The access specifiers of a member controls its accessibility as well as determines the part of the program that can directly access the member of the class.
- In the class declaration syntax, the keywords private, public and protected are known as access specifiers (also known as visibility modes).
- An access specifier specifies the access rules for members following it until the end of the class or until another access specifier is encountered.
- When a member is declared private, it can be accessed only inside the class while a public member is accessible both inside and outside the class. Protected members are accessible both inside the class in which they are declared as well as inside the derived classes of the same class.
- Once, an access specifier has been used, it remains in effect until another access specifier is encountered or the end of the class declaration is reached. An access specifier is provided by writing the appropriate keyword (private, public or protected) followed by a colon ':'.
- Note that the default access specifier of the members of a class is private. i.e., if no access specifier is provided in the class definition, the access specifier is considered to be private.
- In C++ using the access specifier's private, protected, and public, we can set the access of a class's members. We can make members private (visible only to code in the same class), protected (visible only inside the same class and classes derived from it), or public (visible to code inside and outside the class). By default, all the class has private access by default.

- The general **syntax** of a class declaration that uses the private and public access specifiers are given below:

```
class ClassName
{
    private:
        //Declarations of private
        //members appear here ...
    public:
        //Declarations of public
        //members appear here ...
};
```

- For example:

```
class book
{
    //private by default
    char title [30]; //variables declaration
    float price;
    public:
        void getdata(char [],float); //function declaration
        void putdata();
};
```

- In above example, a class named book with two data members title and price and two member functions getdata() and putdata() is created. As no access specifier is provided for data members, they are private by default, whereas, the member functions are declared as public. It implies that the data members are accessible only through the member functions while the member functions can be accessed anywhere in the program.

3.5 DEFINING DATA MEMBERS AND MEMBER FUNCTION [S-22, 23]

3.5.1 Defining Data Member

- All the variables inside the class are called data members. These variables are of any basic data type, derive data type or user-defined data type or objects of other class.
- Any function declare inside a class is called a member function of that class. Only the member functions can have access to the private data members and private functions.

- By default all members of class are private. The general **syntax** of a class declaration is shown below:

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

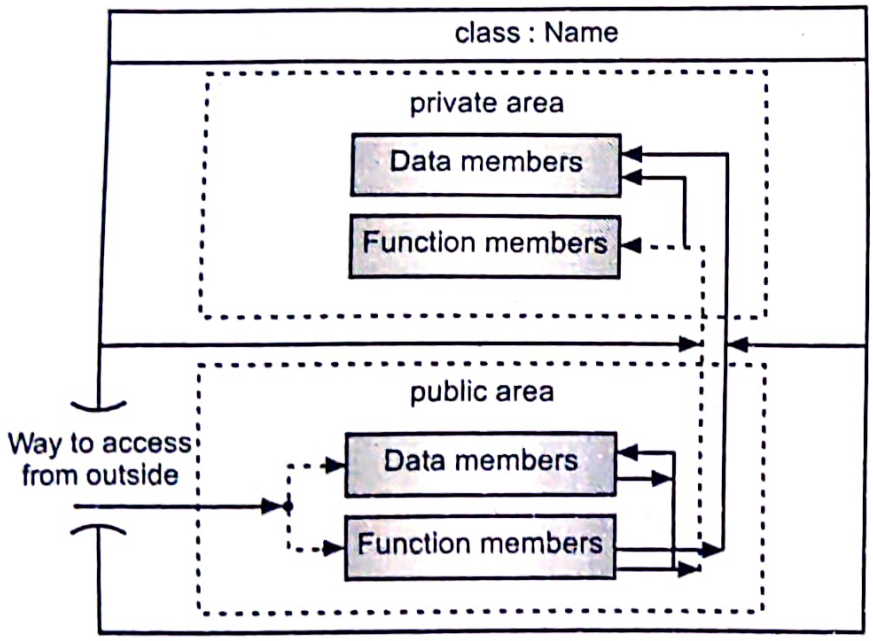

Fig. 3.2: Data Member and Member Function

3.5.2 Defining Member Functions inside and Outside Class Definition [W-22]

- Member functions of a class are defined in following two ways:
  1. Inside the class definition, and
  2. Outside the class.
- In both ways, only syntax of definition of member function changes but body of function (code) is remain same.

1. Inside the Class Definition:

- Functions which are defined inside the class are similar to normal functions and they are handling automatically and inline functions. Normally, functions which are small are defined inside the class.

Example of student class:

```
class student
{
    int roll;
    char name[20];
    public:
        void getdata()          //definition of member function
        {
            cin>>roll>>name;
        }
        void putdata()          //definition of member function
        {
            cout<<roll<<name;
        }
};
```

2. Outside the Class:

- The functions are define after the class declaration, however the prototype of function should be appear inside the class i.e. declaration only (not definition). Since, the function has defined outside the class, there should be some mechanism to know the identity of function to which class they belong we have to use scope resolution operator (::) to identify the function belongs to which class.
- We can also have a function with the same name and same argument list in different classes, since the scope resolution operator will resolve the scope of the function.
- **Syntax:**

```
return type class_name :: function_name (argument list if any)
{
    ..............  //body of function
}
```

- **Example:**

```
class student
{
    ..............
    ..............
    public:
        void getdata(); //declaration of member function
        ..............
        ..............
};
void student::getdata()
{
    ..............
}
```

Program 3.1: Program for class declaration and creation of objects.

```
#include<iostream>
using namespace std;
class date
{
   private:
      int d;
      int m;
      int y;
   public:
      void get (int Day, int Month, int Year);
      void display(); //declaration
};
void date::get (int Day,int Month,int Year)      //definition
{
   d=Day;
   m=Month;
   y=Year;
}
void date::display() //definition
{
   cout<<d<<"-"<<m<<"-"<<y<<endl;
}
int main()
{
   date d1, d2, d3;//date objects d1, d2 & d3
   d1.get (26, 3, 1968);     //call member function
   d2.get(15, 9, 1978);
   d3.get(12, 3, 1992);
   cout<<"Birth Date of the first child:";
   d1.display();
   cout<<"Birth Date of the second child:";
   d2.display();
   cout<<"Birth Date of the third child:";
   d3.display();
}
```

Output:

```
Birth Date of the first child   :  26 - 3 - 1968
Birth Date of the second child  :  15 - 9 - 1978
Birth Date of the third child   :  12 - 3 - 1992
```

Note:

- A member function can call another member function directly.
- No memory is allocated till the point you defined objects of that class.

Const member functions:

- If a member function does not alter or modify any data in the class, in this condition we may declare it as a const member function as:

```
void multi (int, int) const;
double get_balance() const;
```

- The const qualifier is appended to the function prototype.
- A const member function guarantees that it will never modify any of its class's member data.

3.6 SIMPLE C++ PROGRAM USING CLASS

- We usually give class some meaningful name like student.
- This student name now becomes a new type identifier that can be used to declare instances of that class.
- Consider the following simple class example a class called student having roll_no and name as data members.

Program 3.2: Simple C++ program using class.

```
#include <iostream>
#include<cstring>
using namespace std;
class student
{
    int roll_no;
    char name[30];
    public:
        void setdata()
        {
            roll_no = 10;
            strcpy (name, "SAIKRISHNA");
        }
        void getdata()
        {
            cout<<"\n"<<" RollNo:"<<roll_no;
            cout<<"\n"<<"Name:"<<name;
        }
};
int main()
{
    student s;
    s.setdata();
    s.getdata();
}
```

Output:

```
RollNo:10
Name:SAIKRISHNA
```

- Here, the class student contains two data members and two member functions. The data members are private by default where as both the member functions are public by specification.
- The member function setdata() is used to assign values to the data members roll_no and name.
- The member function getdata() is used for displaying the values of data members.
- The data members of the class student can be accessible only to the member functions of a class student.
- It is a general practice to declare data members as private and member functions as public. The general notation of representation of a class student is shown in Fig. 3.3.

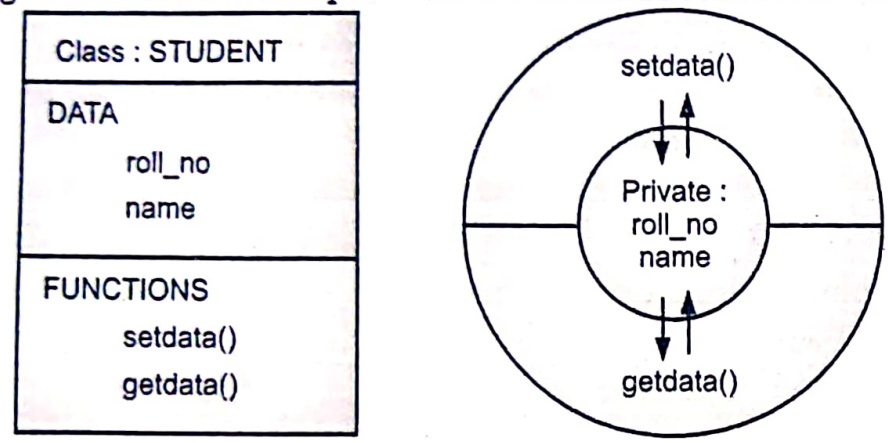

Fig. 3.3: Representation of Class

3.7 MEMORY ALLOCATION FOR OBJECTS [S-22, 23]

- A class is a template from which instances i.e. objects can be created. All the objects created from a class look like and exhibit similar behaviour.
- When a class is declared no storage is allocated for data members. Memory is allocated for objects when they are declared.
- The member functions are created and allocated space in memory only once when they are defined in the class.
- All the objects of that class share the same member functions. But when an object is created, then separate memory space is allocated for the data members of that object, because data members of each object have different values.
- For example, consider the following class:

```
class sample
{
    int x, y;
    public:
        void initialize ( )
        {
            x = 20;
            y = 30;
        }
};
sample s1, s2;
```

- For the above class two objects s1 and s2 are declared. There are two data members in the class x and y which are of integer data type. So total 4 bytes will be allocated for object s1 and 4 bytes for s2.
- Fig. 3.4 illustrates the memory allocation for objects s1 and s2.

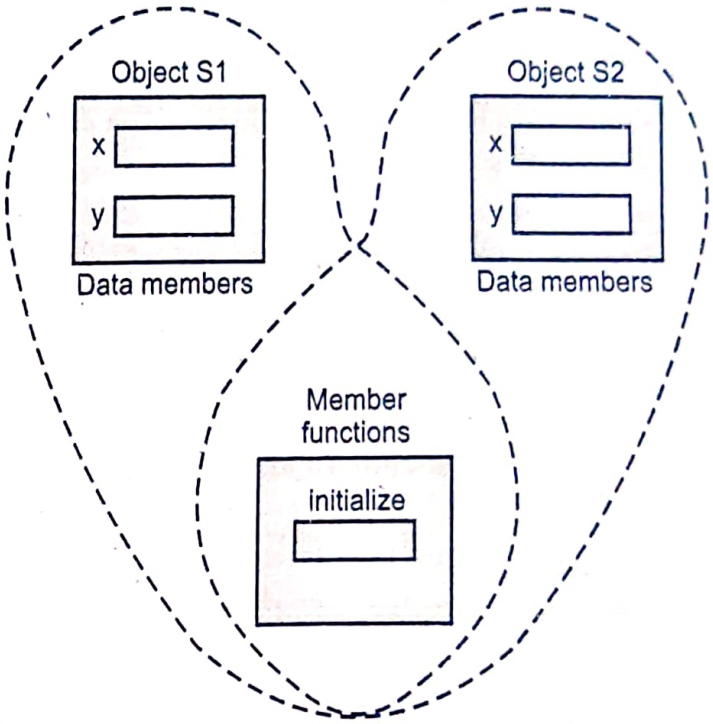

Fig. 3.4: Memory Allocation of Object

- In short the data members are allocated separate space for each object and all objects shares the same member functions.

3.8 STATIC DATA MEMBERS AND STATIC MEMBER FUNCTIONS

[S-22, 23]

- Both that is function and data member of a class can be made static.
- Static variables defined within a function only have a functions scope so that they may not be accessed by any other function.
- C++ also allows us to use static members. The members declared inside the class but persisting from their declaration to the end of program are called static members.
- The static members are both data members as well as member functions.

3.8.1 Static Data Members

[W-18, S-19]

- In C++, a data member of a class can be qualified as static.
- Static variables are normally used to maintain values common to the entire class.
- A static data member is useful when all objects of the same class must share a common item of information.
- Static data members of a class are used to share information among the objects of a class.
- **Syntax for declaring a static data member:**

```
Static datatype data_member_name;
                    OR
datatype static data_member_name;
```

- When you proceed a member variables declaration with *static* you are telling the compiler that one copy of the variable will exist and that of all objects of the class will share that variable.

A static data member has following characteristics:

1. All static variables are automatically initialized to zero value, when the first object of the class is created.
2. Only one copy of static data member is created for the entire class and is shared by all the objects of that class.
3. They are also known as class variables.
4. It is visible only within the class, but its lifetime is the entire program.
5. Static data members are normally used to maintain values common for the all objects.
6. The type and scope of each static variable must be redefined outside the class definition. Because static data members are stored separately rather than as a part of an object.

Program 3.3: Program to read name, post and salary of N employees. Post can be manager or supervisor or worker. Add salary of all managers, supervisors, workers and display it. **[S-18, 22, 23, W-22]**

```
/* Program to illustrate use of static data members in a class */
#include<iostream.h>
#include<conio.h>
#include<string.h>
class EMP
{
    char name [10], post[10];
    int salary;
    public:
    static int m_sum, s_sum, w_sum; \\ variables for 3 posts
    void get_data( )
    {
        cout<< "\n Enter name, post & salary of an employee =";
        cin>>name;
        cin>>post;
        cin>>salary;
    }
    void add_salary( )
    {
        int x;
        x = strcmp(post, "manager");
```

```
        if(x==0)
            m_sum = m_sum + salary;
        x = strcmp(post,"supervisor");
        if(x==0)
            s_sum = s_sum + salary;
        x=strcmp(post,"worker");
        if(x==0)
            w_sum = w_sum + salary;
    }
    void display_data( )
    {
        cout <<"\n"<<name<<post<<salary;
    }
};
// define static variables
int EMP::m_sum;
int EMP::s_sum;
int EMP::w_sum;
void main( )
{
    EMP E[10];
    int N, i, temp;
    clrscr( );
    cout << "\n Enter total no. of employees = ";
    cin >> N;
    for (i=0;i<N;i++)
    {
        E[i].get_data( );
        E[i].add_salary( );
    }
    cout<<"\n Name Post Salary";
    for (i=0;i<N;i++)
    {
        E[i].display_data( );
    }
    cout<<"\n Total salary of all managers="<<EMP::m_sum;
    cout<<"\n Total salary of all supervisors="<<EMP::s_sum;
```

```
cout<<"\n Total salary of all workers="<<EMP::w_sum;
temp=EMP::m_sum + EMP::s_sum + EMP::w_sum;
cout<<"\n Total salary of all the employees in the company = ";
cout << temp;
getch( );
}  // end of main
```

Output:

```
Enter total no. of employees = 4
Enter name, post & salary of an employee =Prajakta manager 10000
Enter name, post & salary of an employee =Prandial supervisor 8000
Enter name, post & salary of an employee =Vaishali worker 5000
Enter name, post & salary of an employee =Megha worker 40000
Name          Post              Salary
Prajakta      manager           10000
Pranjal       supervisor        8000
Vaishali      worker            5000
Megha         worker            4000
Total salary of all managers=10000
Total salary of all supervisors=8000
Total salary of all workers=9000
Total salary of all the employees in the company = 55000
```

- A static variable can be accessed either using object or using the class name. The scope resolution operator must be used when class name is used to access static data.
- By using static member variables, the need for global variables can be eliminated. Since, static variables are associated with the class itself rather than class object; they are also called as class variables.
- In the above program, observe that the static variables are declared within the class, but defined outside the class. This is necessary to emphasize that the memory space for static data is allocated only once before the program starts execution and is shared by entire class. So this is similar to global data.
- The following diagram makes the concept more clear.
- Even if the static data members are private, then also it is necessary to define them outside the class. Also note that the static data members are initialized to zero by default. We can initialize them to any value at the time of definition.

For example: int EMP:: m_sum = 100;

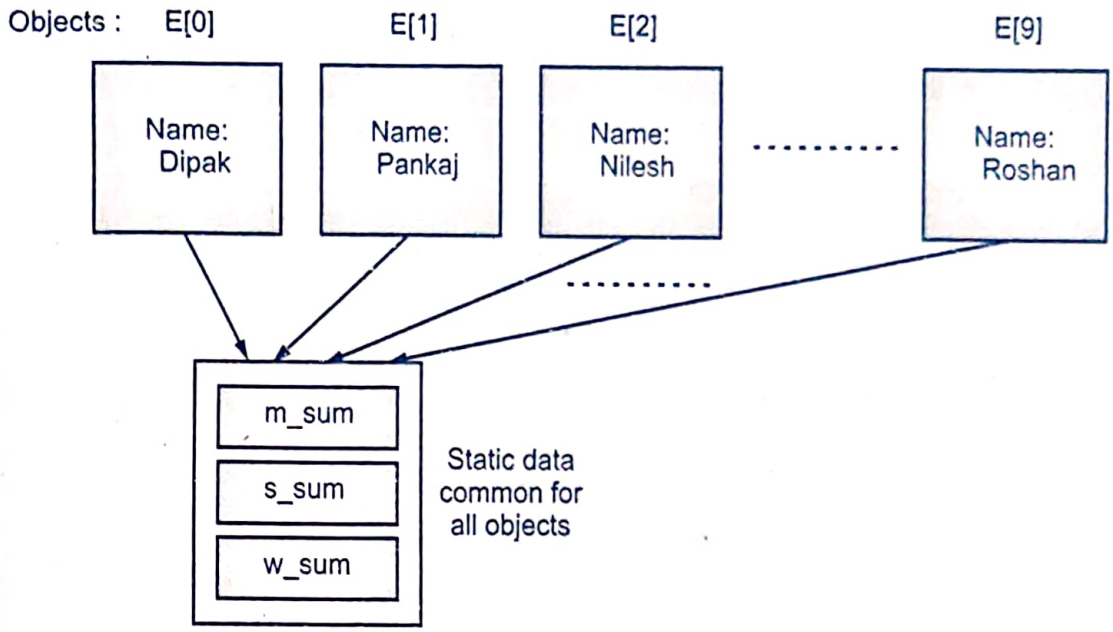

Fig. 3.5: Static Data Members

- This statement will initialize m_sum to 100 instead of zero.
- One more use of static data members is to keep count of total number of objects created for that class.

3.8.2 Static Member Functions

- Member functions may also be declared as static.
- A static function can have access to only other static members declared in the same class.
- But there are some restrictions on them.
  1. They may only access other static members of the same class directly.
  2. They do not have a 'this' pointer.
  3. There cannot be a static and a non-static version of the same function.
- It can be called using class name as,

```
class_name:: function_name
```

- **Syntax of declaring a static member function:**

```
static return_type function_name (arguments);
          OR
return_type static function_name(arguments);
```

- **Syntax for calling static member function:**

```
class_name::member-function-name;
```

Program 3.4: Display a class which displays the number of times display operation is performed. [W-18]

```
#include<iostream.h>
class display
{
   private:
      static int count;
```

```
    public:
        static void disp()
        {
            count++; //increment cout
            cout<<count<<endl;
        }
};
int display:: count; //defining static variable outside a class
int main()
{
    display ob1,ob2,ob3,ob4;
    ob1.disp();
    ob2.disp();
    ob3.disp();
    ob4.disp();
    //function call with class name
    display:: disp();
}
```

Output:

```
1
2
3
4
5
Press any key to continue . . .
```

- The statement count++ is executed when disp() function is invoked and the current value of count is assigned to each object.

3.9 ARRAY OF OBJECTS [S-19, 22]

- The array of variables of class data type is called Array of Objects.
- An array of objects is an array of a class type is also known as an array of objects. An array of objects is declared in the same way as an array of any built-in data type.
- The **syntax** for declaring an array of objects is as follows:

```
class_name array_name[size];
```

- For example: If class is student and we want to define that there are 100 students in class. So we define array of student with 100 objects and here student is object of class type. The size of an object is defined from class declaration. So the storage required for array of object is defined by complex from class declaration.

Example:

```
class student
{
    int roll;
    char name[20];
    public:
        void getdata();
        void putdata();
};
```

If we create object of class student

```
student s; // single object
```

The array of objects is created as:

```
student s[100]; //array of student
```

Here, the array of s contains 100 objects.

We an also have the array of different categories of the student,

```
student sc[10];
student open[30];
student obc[20];
```

- The array of **sc** contains 10 objects, the array of open contains 30 objects, and the array of **obc** contains 20 objects. The behaviour and attributes of each object is define is class where the identifier name is same but values are different.
  For example, Identifier roll has values 100, 101 .....
- The array of object is stored in memory as same as how multidimensional array are stored. The space is required only for data item of the class. The array of student is represent in Fig. 3.6.

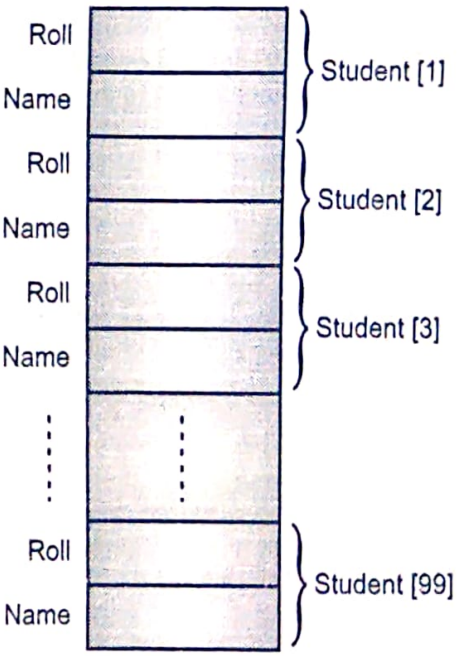

Fig. 3.6: Storage for Data Items in Array of Objects

- We have discussed how single object access the data items on member function. In same way, using array of objects we can access members of class (using dot).
  For example, `student[i].getdata();`
  Here, we can input the i^{th} element data of student array.
  For example, `student[i].putdata();`
  Hence, we can display the i^{th} element of the student array.

Program 3.5: Program for array of objects. [S - 18]

```
#include<iostream>
using namespace std;
class student
{
   int roll;
   char name[20];
   public:
      void getdata (void);
      void putdata (void);
};
void student:: getdata(void)
{
   cout<<"Enter roll:";
   cin>>roll;
   cout<<"Enter name:";
   cin>>name;
}
void student::putdata(void)
{
   cout<<"roll:"<<roll<<endl;
   cout<<"name:"<<name<<"\n";
}
int main()
{
   int i;
   student s[3]; //array of size three
   for(i=0;i<3;i++)
   {
      cout<<i+1<<"Student Information"<<endl;
      s[i].getdata();
   }
   for (i=0;i<3;i++)
   {
      s[i].putdata();
   }
   return 0;
}
```

Output:

```
1 Student Information
   Enter roll: 200
   Enter name: Nirali
```

```
2 Student Information
    Enter roll: 201
    Enter name: Prajakta
3 Student Information
    Enter roll: 202
    Enter name: Anita
    roll: 200
    name: Nirali
    roll: 201
    name: Prajakta
    name: 202
    name: Anita
```

3.10 OBJECTS AS FUNCTION ARGUMENT [S-22, 23]

- In C++ an object may be used as a function argument.
- The following two approaches used in objects as a function argument.

1. Passing Object to function:

- Objects may be passed to a function just like any other variable. Arguments can be passed to a function in two ways:
  - Call by value (A copy of entire object is passed to the function)
  - Call by reference (Only the address of the object is transferred to the function).
- Objects are passed through the call by value mechanism by default. In this method value of the variable (object) will be passed to function as an argument or parameter.

2. Returning objects from the function:

- A function may return an object to the calling function.
- It works similar to the normal integer or float variables. For such functions return type will be name of a class.

Program 3.6: Program for object as function argument concept.

```
/* Program which adds 2 complex numbers */
#include<iostream.h>
#include<conio.h>
class COMPLEX
{
    int real, imag;
    public:
        void get_no(int a, int b)
        {
            real = a;
            imag = b;
        }
```

```
        void display_no( )
        {
            cout<<"\n Real ="<<real;
            cout<<"\n Imag = "<<imag;
        }
        COMPLEX add_no (COMPLEX C2)
        {
            COMPLEX C3;
            C3.real = real + C2.real;
            C3.imag = imag + C2.imag;
            return(C3);
        }
};  // end of class
int main( )
{
    COMPLEX C1, C2, C3;
    C1.get_no(10, 20);
    C2.get_no(30, 40);
    C3=C1.add_no(C2);
    cout<<"\n Addition";
    C3.display_no( );
}
```

Output:

```
Addition
Real = 40
Imag = 60
Press any key to continue . . .
```

3.11 FRIEND FUNCTION [S-19, 23, W-23]

- The functions that are declared with the keyword friend are known as friend functions.
- To make an outside function friendly to a class, we have to simply declare this function as a friend of the class.
- A friend function is a function that is not a member of a class but has access to the class's private and protected members. Friend functions are not considered class members; they are normal external functions that are given special access privileges.
- Friends are not in the class's scope, and they are not called using the member selection operators (. and –>) unless they are members of another class.

- **For example:**

```
class xyz
{
    ……
    public:
    ……
    ……
    friend void abc(void);
};
```

- The function declaration in above example should be preceded by the keyword friend.

Program 3.7: C++ Program to Add Two Numbers using Friend Function. [S-19]

```
#include<iostream>
using namespace std;
class temp
{
  int x, y, z;
  public:
    void input()
    {
        cout << "Enter the value of x and y:";
        cin >> x>>y;
    }
    friend void add(temp &t);
    void display()
    {
        cout << "The sum is :" << z;
    }
};
void add(temp &t)
{
   t.z = t.x + t.y;
}
int main()
{
   temp t1;
   t1.input();
   add(t1);
   t1.display();
   return 0;
}
```

Output:

```
Enter the value of x and y:10 20
The sum is :30
```

3.11.1 Friend Class

- A friend class is a class all of whose member functions are friend functions of a class, that is, whose member functions have access to the other class's private and protected members.
- A friend class in C++, can access the "private" and "protected" members of the class in which it is declared as a friend. On declaration of friend class all member function of the friend class become friend of the class in which the friend class was declared. Friend status is not inherited; every friendship has to be explicitly declared.
- Classes are declared as friends within the definition of the class to whom access is to be given; this prevents a class from giving itself access to another's protected members, which enforces encapsulation.
- The friend class has the same level of access irrespective of whether the friend declaration appears in either the public, protected or private sections of the class definition. Friend status is granted by using the friend keyword.

```
friend class ClassName;
```

Program 3.8: Program illustrate friend class concept.

```
#include <iostream.h>
#include <conio.h>
class B
{
  // B declares A as a friend...
  friend class A;
  private:
     void privatePrint()
     {
        std::cout << "hello, world" << std::endl;
     }
};
class A
{
  public:
      A()
      {
         B b;
         // ... and A now has access to B's private members
         b.privatePrint();
      }
};
int main()
{
 A a;
 return 0;
}
```

Output: hello, world

Advantages of using friend class:

1. It provides additional functionality which is kept outside the class.
2. It provides functions with data which is not normally used by the class.
3. It allows sharing private class information by a non member function.

3.12 FUNCTION RETURNING OBJECTS

- A function may return an object to the calling function.
- It works similar to the normal integer or float variables. For such functions return type will be name of a class.

Program 3.9: Program for returning objects.

```
/* Program which adds 2 complex numbers */
   #include<iostream.h>
   #include<conio.h>
   class COMPLEX
   {
      int real, imag;
      public:
      void get_no(int a, int b)
      {
         real = a;
         imag = b;
      }
      void display_no( )
      {
         cout<<"\n Real ="<<real;
         cout<<"\n Imag = "<<imag;
      }
      COMPLEX add_no (COMPLEX C2)
      {
         COMPLEX C3;
         C3.real = C1.real + C2.real;
         C3.imag = C1.imag + C2.imag;
         return(C3);
      }
   }; // end of class
void main( )
{
   COMPLEX C1, C2, C3;
   C1.get_no(10, 20);
   C2.get_no(30, 40);
   C3=C1.add_no(C2);
   cout<<"\n Addition =";
   C3.display_no( );
}
```

Output:

```
Addition
real = 40
imag = 60
```

Program 3.10: Program to display students data.

```
#include<iostream>
using namespace std;
class student
{
   int marks[5];
   int total=0;
   char name[20];
   public:
      void read_data();
      void display_data();
};
void student:: read_data()
{
   int i;
   cout <<"Enter student name:";
   cin >> name;
   cout <<"Enter 5 subject marks:";
   for (i = 0; i < 5; i++)
   {
      cin >> marks[i];
      total = total + marks[i];
   }
}
void student:: display_data()
{
   cout <<"\n Name:"<< name;
   cout <<"\n Marks in 5 subjects:";
   for (int i = 0; i < 5; i++)
   cout <<"\t" << marks[i];
   cout <<"Total ="<< total;
}
```

```
int main()
{
    student s;
    s.read_data();
    s.display_data();
    return(0);
}
```

Output:

```
Enter student name:Rekha
Enter 5 subject marks:20
50
30
60
40
Name:Rekha
Marks in 5 subjects:   20 50 30 60 40 Total = 200
```

Program 3.11: Program find maximum of two numbers.

```
#include<iostream>
using namespace std;
class number
{
    int a, b;
    int find_max();
    public:
        void init(int, int);
        void disp_max();
};
void number:: init (int x, int y)
{
    a = x;
    b = y;
}
int number:: find_max()
{
    if (a > b)
        return a;
    else
        return b;
}
```

```
void number:: disp_max()
{
    cout <<"\n Two numbers are";
    cout << a << "and" << b;
    cout <<"Maximum number is "<< find_max();
}
int main()
{
    number n;
    n. init (10, 20);
    n. disp_max();
    return(0);
}
```

Output:

```
Two numbers are 10 and 20
Maximum number is 20.
```

Program 3.12: Program to declare class account having data member's principle, rate_of_interest, no_of_years. Accept this data for one object and find out simple interest. [S-23]

```
#include<iostream>
using namespace std;
class account
{
   private:
      int p, n, r;
      float si;
   public:
      void getdata()
      {
         cout <<"Enter Principle, Rate_of_interest and number_of_years:\n";
         cin >> p >> r >> n;
      }
      void calculate ()
      {
         si = (p * n * r) / 100;
      }
      void display()
      {
         cout <<"Simple Interest =";
         cout << si;
      }
};
```

```
int main()
{
    account a;
    a.getdata ();
    a.calculate();
    a.display();
    return(0);
}
```

Output:

```
Enter Principle, Rate_of_interest and number_of_years:
5400 12 5
Simple Interest = 3240
```

Program 3.13: Program to declare a class rectangle having data member's length and breadth. Accept this data for one object and display area and perimeter of rectangle.

```
#include<iostream>
using namespace std;
class rectangle
{
   int len, bred, area, peri;
public:
   void accept( );
   void calculate( );
   void display( );
};
void rectangle:: accept()
{
   cout <<"Enter length and breadth:";
   cin >> len >> bred;
}
void rectangle:: calculate()
{
   area = len * bred;
   peri = 2 * (len + bred);
}
void rectangle:: display()
{
   cout<<"Area of rectangle:" << area;
   cout<<"\n Perimeter of Rectangle:"<<peri;
}
```

```
int main( )
{
    rectangle r1;
    r1.accept( );
    r1.calculate( );
    r1.display( );
    return(0);
}
```

Output:

```
Enter length and breadth:9 5
Area of rectangle:45
Perimeter of Rectangle:28
```

Program 3.14: Write a C++ program to find reverse of a number using friend function.

```
#include<iostream>
using namespace std;
class T4Tutorials
{
    private:
    int n,i;
     public:
        T4Tutorials()
        {
            cout<<"Enter Number to Display reverse: ";
            cin>>n;
        }
        friend void show(T4Tutorials);
};
void show(T4Tutorials r)
{
    cout<<"The reverse the Entered number: ";
    for(r.i=r.n;r.i>0;r.i=r.i/10)
    {
        cout<<r.i%10;
    }
}
```

```
int main()
{
    T4Tutorials r;
    show(r);
}
```

Output:

```
Enter Number to Display reverse: 1234
The reverse the Entered number: 4321
```

Program 3.15: Program to display book information using class object.

```
#include<iostream>
using namespace std;
class book
{
   int bookno;
   char bookname[20];
   char author[20];
   float price;
   public:
      void getdata();
      void display();
};
void book::getdata()
{
   cout<<"enter book number";
   cin>>bookno;
   cout<<"enter book name";
   cin>>bookname;
   cout<<"enter author name";
   cin>>author;
   cout<<"enter book price";
   cin>>price;
}
void book::display()
{
   cout<<"book number"<<bookno<<endl;
   cout<<"book name="<<bookname<<endl;
```

```
    cout<<"author="<<author<<endl;
    cout<<"price="<<price<<"Rs.";
}
int main()
{
    book b;
    b.getdata();
    b.display();
}
```

Output:

```
Book no = 2503
Book name = C++
Author = Bharambe
Price = 99.99 Rs.
```

Summary

- C++ structures and classes combines/packs data and the functions together into a single entity. However, to follow the conventions of object-oriented programming, the classes are generally used to contain both data and functions, and the structures are generally used to contain only data.
- Objects are instances of class, which holds the data variables declared in class and the member functions work on these class objects.
- Classes are the central feature of C++ that supports OOP and are often called user defined types.
- An object is a real word thing which performs a specific task. A class serves as a blueprint or template for its objects. That is, once a class has been defined, any number of objects belonging to that class can be created. The objects of a class are also known as the instances.
- A class definition starts with the keyword class followed by the class name; and the class body enclosed by a pair of curly braces.
- The variables declared in the class are known as data members, while the functions declared in the class are known as member functions.
- The keywords private, public and protected are known as access specifiers, Which restrict or limits the accessibility of class members.
- A public member is accessible from anywhere outside the class but within a program.
- A private member variable or function cannot be accessed, or even viewed from outside the class. Only the class and friend functions can access private members. By default all the members of a class would be private.

- A protected member variable or function is very similar to a private member but it provided one additional benefit that they can be accessed in child classes which are called derived classes.
- A class provides the blueprints for objects, so basically an object is created from a class.
- A class is only a logical abstraction that specifies what data and functions its objects will have, whereas the objects are the physical entities through which those data and functions can be used in a program.
- The main difference between a structure and a class in C++ is that, the data and functions in a structure are by default public, whereas the data and functions in a class are by default private.
- While defining a member function outside the class, function name in the function header is preceded by the class name and the scope resolution operator (: :).
- The member functions defined inside a class definition are by default inline functions.
- Like array of other user-defined data types, an array of type class can also be created. The array of type class contains the objects of the class as its individual elements. Thus, an array of a class type is also known as an array of objects. An array of objects is declared in the same way as an array of any built-in data type.

Check Your Understanding

1. An array of class type is known as .....
   (a) array of objects (b) pointers to array
   (c) array of pointers (d) both (a) & (b)
2. By default members of structures are .....
   (a) protected (b) private
   (c) public (d) all of above
3. Which of the following keywords is used to control access to a class member?
   (a) default (b) break
   (c) protected (d) goto
4. ..... keyword used for declared an structure in C++.
   (a) class (b) struct
   (c) private (d) protected
5. Identify correct access specifiers in following:
   (a) public, private, friend (b) public, protected, this
   (c) public, private, protected (d) none of above
6. Objects are instances of class.
   (a) True (b) False

7. A member function can call another member function directly without using dot operator.
   (a) True (b) False

Answers

1. (b)	2. (c)	3. (c)	4. (b)	5. (c)	6. (a)	7. (a)

Practice Questions

Q.I Answer the following questions in short:

1. What is a class? Give its syntax.
2. What are objects? Give its syntax.
3. How to create Object and class?
4. What is friend function?
5. How to define class member and member function give simple example?

Q.II Answer the following questions:

1. How to create class and object? Explain with example.
2. How memory is allocated when multiple objects of a class are created? Explain with suitable example.
3. What is meant by class members?
4. What is access specifier? Enlist them.
5. Explain class members in detail.
6. Write a program for processing objects of the student class. Declare member functions such as show() as read-only member functions.
7. Define a class to represent a bank account. Include the following members:
   Data members
   (a) Name of the depositor
   (b) Account number
   (c) Type of account
   (d) Balance amount in the account.
   Member functions:
   (a) To assign initial values
   (b) To deposit an amount (bal = bal + amt)
   (c) To withdraw an amount after checking balance (bal = bal – amt)
   (d) To display name and balance.
   Write a main program to test the program.
8. Write a C++ program to display n number of employees with information such as employee number, employee name and employee salary.

9. Write a C++ program to illustrate string operations: length, append, erase, substr, replace, front, back and at() operations.
10. Write a C++ program to accept student information as sno, sname, sub1 and sub2 for five students using array of objects. Calculate total marks and display output.
11. Describe array of objects in detail.
12. Compare class and object.
13. How to define class members and member functions? Explain with example.

Q.III Define the term:

1. Class
2. Object
3. Data Member
4. Member function
5. Friend class

Previous Exam Questions

April 2018

1. Create a class student having the following members:

- Rollno
- Name
- Percentage

Write necessary member function to accept student details and display details along with class obtained depending on percentage. **[4 M]**

Ans. Refer to Program 3.5.

2. Design C++ class which contains function display( ). Write a program to count number of times display( ) function is called. (Use static data member). **[4 M]**

Ans. Refer to Program 3.3.

October 2018

1. Write a C++ program to find reverse of a number using friend function. **[4 M]**

Ans. Refer to Program 3.4.

2. What is static data member? Explain its characteristics. **[4 M]**

Ans. Refer to Section 3.8.1.

April 2019

1. What is static data member? **[2 M]**

Ans. Refer to Section 3.8.1.

2. Explain array of object with diagram. **[4 M]**

Ans. Refer to Section 3.9.

3. Write a C++ program using friend function to calculate sum of digits of a number. [4 M]

Ans. Refer to Program 3.7.

4. What is friend function? Which are the features of friend function? [4 M]

Ans. Refer to Section 3.11.

5. Trace the output of the following program and explain it. Assume there is no syntax error. [4 M]

```
# include <iostream>
using namespace std;
int i, j;
class SYBCA
{
   public :
   SYBCA(int x=0, int y=0)
   {
      i=x;
      j=x;
      Display();
   }
   void Display()
   {
      cout <<j<<" " ;
   }
};
int main()
{
   SYBCA obj(10, 20);
   int &s=i;
   int &z=j;
   i++;
   cout << S - - << " " << ++z;
   return 0;
}
```

Output:

```
10 11 11
```