Chapter-7

CLASSES AND OBJECTS

➢ Classes:

• A class is a collection of objects that have identical properties, common behavior and shared relationship.
• A class binds the data and its related functions together.

➢ Definition and Declaration of Classes:

• A class definition is a process of naming a class and data variables, and interface operation of the class.
• The variables declared inside a class are known as data members.
• The functions declared inside a class are known as member functions.
• A class declaration specifies the representation of objects of the class and set of operations that can be applied to such objects.
• The general syntax of the class declaration is:

```cpp
class User_Defined_Name
{
    private :
        Data Member;
        Member functions;
    public :
        Data Member;
        Member functions;
    protected :
        Data Member;
        Member functions;
};
```

• Key word 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.
• Example: Let us declare a class for representation of bank account.

```cpp
class account
{
    private:
        
```
```cpp
int accno;
char name[20];
char acctype[4];
int bal_amt;

public:
    void get_data();
    void display_data();
};
```

### Access Specifiers:

- Every data member of a class is specified by three levels of access protection for hiding data and function members internal to the class.
- They help in controlling the access of the data members.
- Different access specifiers such as private, public, and protected.

**private:**

- **private** access means a member data can only be accessed by the class member function or friend function.
- The data members or member functions declared **private** cannot be accessed from outside the class.
- The objects of the class can access the private members only through the public member functions of the class. This property is also called *information hiding*.
- By default, data members in a class are private.
- Example:
  ```cpp
  private:
      int x;
      float y;
  ```

**protected:**

- The members which are declared using **protected** can be accessed only by the member functions, friend of the class and also the member functions derived from this class.
- The members cannot be accessed from outside the class.
- The **protected** access specifier is similar to private access specifiers.

**public:**

- **public** access means that member can be accessed any function inside or outside the class.
- Some of the **public** functions of a class provide interface for accessing the private and protected members of the class.
Member Function:

- Member functions are functions that are included within a class (Member functions are also called Methods).

- Member functions can be defined in two places.
  - Inside class definition
  - Outside class definition

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.

Example:

```cpp
class rectangle
{
    int length, breadth, area;
    public:
        void get_data()
        {
            cout<< " Enter the values for Length and Breadth";
            cin>>length>>breadth;
        }
        void compute()
        {
            area = length * breadth;
        }
        void display()
        {
            cout<<" The area of rectangle is"<<area;
        }
};
```
Outside class definition:

- To define member function outside the class declaration, you must link the class name of the class with the name of member function.
- We can do this by preceding the function name with the class name followed by two colons (::).
- The two colons (::) are called scope resolution operator.
- Scope resolution operator (::) is used to define the member function outside the class.
- The general form of a member function defined outside the class is:

```
return_type class_name : : member_function_name( arg1, arg2, ….argN)
{
    function body;
}
```

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

Program: To use classes using member functions inside and outside class definition.
```cpp
#include<iostream.h>
class item
{
    private:
        int numbers;
        float cost;
    public:
        void getdata(int a, float b);
        void putdata( )
        {
            cout<<"Number: "<<number<<endl;
            cout<<"Cost:"<<cost<<endl;
        }
};
```
### Defining object of a class:

- An object is a real world element which is identifiable entity with some characteristics (attributes) and behavior (functions).
- An object is an instance of a class. Objects are sometimes called as instance variables.
- An object is normally defined in the main () function.
- The syntax for defining objects of a class as follows:

  ```
  class Class_Name
  {
  private :  //Members
  public :   //Members
  }
  
  class Class_Name Object_name1, Object_name2, ……;
  ```

  where class keyword is optional.

- **Example 1:** The following program segment shows how to declare and create objects.

  ```
  class Student
  {
  private:
  int rollno;
  char name[20];
  char gender;
  int age;
  
  public:
  void get_data( );
  void display( );
  
  }
  
  Student S1, S2, S3;     //creation of objects
  ```

- Here, creates object S1, S2, and S3 for the class Student.
- When an object is created space is set aside for it in memory.

```c
void item::getdata(int a, float b)
{
    number = a;
    cost = b;
}
int main()
{
    item x;
    x.getdata(250, 10.5);
    x.putdata();
    return 0;
}
```

**OUTPUT:**

Number: 250  
Cost: 10.5
• Example 2:

```cpp
class num
{
private:
    int x, 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);
}
```

Accessing member of the class:

• The member of the class can be data or functions.
• Private and protected members of the class can be accessed only through the member functions of the class.
• No functions 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 of accessing member (data and functions) of a class is:
  a) Syntax for accessing a data member of the class:

    `Object_Name . data_member;`
  b) Syntax for accessing a member function of the class:

    `Object_Name . member_function(arguments)`
Example:

class rectangle
{
    int length, breadth, area;
    public:
        void get_data()
        {
            cout<<"Enter the length and breadth"<<end;
            cin>>length>>breadth;
        }
        void compute()
        {
            area = length * breadth;
        }
        void display()
        {
            cout<<"The area of rectangle is "<<area;
        }
};
void main()
{
    rectangle r1;
    clrscr();
    r1.get_data();
    r1.compute();
    r1.display();
    getch();
}

Array as member of classes:
- Array can be used as a data member of classes.
- An array can be used as private or public member of a class.
- This is illustrated in the following program.

```
#include<iostream.h>
#include<conio.h>
class array
{
    private:
        int a[100], m;
    public:
        void setnoofelements(int n);
        { m = n; }
```
void readarray( );
void displayarray( );

void array : : readarray( )
{
    cout<<"Enter "<<m<<" Array elements"<<endl;
    for(int i=0; i<m; i++)
        cin>> a[i];
}
void array : : displayarray( )
{
    cout<<"Array elements are:"<<endl;
    for(int i=0; i<m i++)
        cout<< a[i]<<endl;
}
void main( )
{
    int n;
    array a;
    clrscr( );
    cout<<"Input number of elements:"<<endl;
    cin>>n;
    a.setnoofelements(n);
    a.readarray( );
    a.dispalyarray( );
    getch( );
}

### Classes, Objects and Memory:

- The class declaration does not allocate memory to the class data member.
- When a object is declared, memory is reserved for only data members and not for member functions.

### The following program illustrates as follows:

```cpp
#include<iostream.h>
#include<conio.h>
class student
{
    private:
        long regno; // 4 bytes of memory
        char name[20]; // 20 bytes of memory
        char comb[4]; // 4 bytes of memory
        int marks; // 2 bytes of memory
```

---

**OUTPUT:**

Input number of elements: 5
Enter 5 Array elements
10 20 30 40 50
Array elements are:
10 20 30 40 50
char address[30]; // 30 bytes of memory

public:
    void readdata( );
    void display( );
);
void main( )
{
    student s1, s2;
    cout<<"Size of the object s1 is = "<<sizeof(s1)<<endl;
    cout<<"Size of the object s2 is = "<<sizeof(s2)<<endl;
    cout<<"Size of the class is = "<<sizeof(student)<<endl;
}

➤ Array of Objects:

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

```cpp
class employee
{
    private:
        char name[10];
        int age;
    public:
        void readdata( );
        void displaydata( );
};
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].
- The storage of data items in an object array is:

<table>
<thead>
<tr>
<th></th>
<th>Name</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>supervisor[0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>supervisor[1]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>supervisor[2]</td>
<td></td>
</tr>
</tbody>
</table>

- Program to show the use of array of objects:

```cpp
#include<iostream.h>
#include<conio.h>
class data
{
    private:
```
int regno, maths, computer;
public:
    void readdata( );
    void average( );
    void display( );
};
void data::readata( )
{
    cout<<"Enter Register No:";
    cin>>regno;
    cout<<"Enter Maths marks:";
    cin>>maths;
    cout<<"Enter Computer marks:";
    cin>>computer;
    display( );
}
void data::average( );
{
    int avg;
    avg = (maths+computer)/2;
}
void data::display( )
{
    cout<<"Average = "<<average( )<<endl;
}
void main( )
{
    data stude[3];
    clrscr( );
    for(i=0; i<3; i++)
    {
        stude[i].readdata( );
        getch( );
    }

Objects as function arguments:

- A function can receive an object as a function argument.
- This is similar to any other data 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**, copy of 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.
private:
    float phy, che, mat;
public:
    void readdata() {
        cout<<"Input Physics, Chemistry, Maths marks : " ;
        cin>>phy>>che>>mat;
    }
    void total(exam PU, exam CT) {
        phy = PU.phy + CT.phy;
        che = PU.che + CT.che;
        mat = PU.mat + CT.mat;
    }
    void display() {
        cout<< "Physics :" <<phy<<endl;
        cout<< "Chemistry :" <<che<<endl;
        cout<< "Maths :" <<mat<<endl;
    }
};
void main( ) {
    Exam PUC, CET, Puc_plus_Cet;
    clrscr( );
    cout<<"Enter PUC Marks"<<endl;
    PUC.readdata( );
    cout<<"Enter CET Marks"<<endl;
    CET.readdata( );
    Puc_plus_Cet.total(PUC, CET);
    cout<<"Total marks of PUC and CET is:" <<endl;
    Puc_plus_Cet.display( );
}

• 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 requires only passing the address of the object and not the entire object.

➢ Difference between Structure and Classes:

<table>
<thead>
<tr>
<th>Structure</th>
<th>Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>A structure is defined with the struct keyword</td>
<td>A class is defined with the class keyword</td>
</tr>
<tr>
<td>All the member of a structure are public by default</td>
<td>All the members of a class are private by default</td>
</tr>
</tbody>
</table>

OUTPUT:

Enter PUC Marks
Input Physics, Chemistry, Maths marks : 67 89 80
Enter CET Marks
Input Physics, Chemistry, Maths marks : 60 76 91
Total marks of PUC and CET is:
Physics: 127
Chemistry: 165
Maths: 171
### IMPORTANT QUESTIONS:

**1 Mark questions:**

1. What is a Class, Objects, Data Member, Member Functions, Scope Resolution Operator, and Array of objects?
2. Mention the access specifiers used with a class?

**5 Mark questions:**

1. Explain class definitions and class declaration with syntax and example.
2. Explain Member function.
   - a. Inside class definition
   - b. Outside class definition
3. Explain the array of objects.

#### Exercise programs

1. Write a C++ program to find the simple interest using class and objects.

```cpp
#include<iostream.h>
#include<conio.h>
class SI
{
    private:
        float p, t, r, si;
    public:
        void readdata()
        {
            cout<<"Enter the Principal Amount, Time & Rate"<<endl;
        }
};
```
cin>>p>>t>>r;
} 
void compute( ) 
{ 
    si = (p * t * r)/100; 
} 
void display( ) 
{ 
    cout<<"Simple Interest = "<<si; 
}; 
void main( ) 
{ 
    SI s; 
    clrscr( ); 
    s.readdata( ); 
    s.compute( ); 
    s.display( ); 
    getch( ); 
}

2. Let product list be a linear array of size N where each element of the array contains 
following field Itemcode, Price and Quantity. Declare a class Product list with three data 
members and member functions to perform the following

a. Add values to the product list
b. Printing that total stock values

#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
class product 
{ 
    private: 
        char itemcode[6];
        float price, quantity;
    public:
        void Addproduct( )
        { 
            cout<<"Enter the Item Code"<<endl;
            cin>>itemcode;
            cout<<"Enter the Price"<<endl;
            cin>>price;
            cout<<"Enter the Quantity"<<endl;
            cin>>quantity;
        }
```cpp
void display() {
    cout << itemcode << " \n" << price << " \n" << quantity << endl;
}
}

void main() {
    int N = 0;
    char ans;
    product list[100];
    clrscr();
    while(1) {
        cout << "Item Code, Price and Quantity" << endl;
        List[N].AddProduct();
        cout << "Do you want to add next item (Y/N)?" << endl;
        cin >> ans;
        if (toupper(ans) == 'N')
            break;
        N++;
    }
    cout << "Item Code \n Price \n Quantity" << endl;
    for (i = 0; i < N; i++)
        List[i].display();
    getch();
}

3. A class clock has following member hours and minutes. Create member function
a. To initialize the data members
b. Display the time
c. To convert hours and minutes to minutes.
```
```
cout<<"Hours = "<<hh;
cout<<"Minutes = "<<mm;
}
void convert()
{
    mm = hh * 60 + mm;
cout<<"Total Minutes = "<<mm;
}
};
void main()
{
    int h, m;
    clock c;
    clrscr();
cout<<"Enter the Hour and Minutes"<<endl;
cin>>h>>m;
c.initialize();
c.display();
c.convert();
getch();
}

4. Write a C++ program that receives arrival time and departure time and speed of an automobile in kilometers/hours as input to a class. Compute the distance travelled in meters/second and display the result using member functions.
#include<iostream.h>
#include<conio.h>
class Distance
{
    private:
        int Ahh, Amm, Dhh, Dmm;
        float speed;
    public:
        void inputtime()
        {
            cout<<"Enter the Arrival Time:"<<endl;
cout<<"Enter the Hour and Minutes"<<endl;
cin>>Ahh>>Amm;
cout<<"Enter the Departure Time:"<<endl;
cout<<"Enter the Hour and Minutes"<<endl;
cin>>Dhh>>Dmm;
cout<<"Enter the speed in Kmph"<<endl;
cin>>speed;
        }
        void computedistance()
{ 
    float dist;
    dist = ( (Ahh * 60 + Amm) - (Dhh * 60 + Dmm) ) * speed/60;
    dist = (dist * 1000 / (60 * 60));
    cout<<"Distance Travelled = "<<dist<<"Meter/Second";
}
};

void main( )
{
    Distance d;
    clrscr( );
    d.inputtime( );
    d.computedistance( );
    getch( );
}