CLASS AND OBJECTS

Chapter 3

An Introduction to classes

- A class is a building block of OOP. It is the way to bind the data and its logically related functions together.
- An abstract data type that can be treated like any other built in data type.

Class definition

- Class head → class name_of_class.
- Class body →

```
 {
  data members;
  member functions;
};
```

Example

```
Class test
{
private :
        int a;
        int b;
public:
        void set_data(int x, int y)
        {
                a=x;
                b=y;
        }
        int  big()
        {
        if (a > b)
                return a;
        else
                return  b;
        }
};
```

Characteristics of access specifiers (private,public and protected)

- Private section of a class can have both data members and member functions, usually data members are made private for data security.
- It is not mandatory that private section has to declared first in the class and then the public section.
- If no member access specifier is specified then by default the members are private for the class.
- There may be any number of private, public or protected section in a class declaration.
- Protected specifier is used for declaring the class members which can be accessed by its own class and its derived class.

Member Function

- Member function’s name is visible outside the class.
- It can be defined inside or outside the class.
- It can have access to private, public and protected data members of its class, but cannot access private data members of another class.

Introduction to objects

- Object is an abstraction of real wold entity.
- Objects are the variables/instances of classes.
- Syntax for declaring objects is as follows :

  <class name>
  <obj_name1>,<obj_name2>, …, <obj_name1>
  ;

Example: for class test the objects can be created as follows:

test t1,t2,…tn;

Characteristics of objects:

- It can have its own copy of data members.
- The scope of an object is determined by the place in which the object is defined.
- It can be passed to a function like normal variables.
- The members of the class can accessed by the object using the object to *member access operator* or *dot operator(.).*

Example:

```
Class test
{
private :
        int a;
        int b;
public:
        void set_data(int x, int y)
        {
                a=x;
                b=y;
        }
        int  big()
        {
        if (a > b)
                return a;
        else
                return  b;
        }
};
void main()
{
test t;
int a,b;
cout<<“enter the two numbers” << endl;
cin>> a >> b;
t.set_data(a,b);
cout<<“the largest number is ” << t.big() << endl;
}
```

Definition of function outside a class.

- The syntax for defining function outside a class is as follows:

<data type> <class name> :: <function name> ()

{

// member function definition

}

Data type → return type

Class name → the class to which the member function belongs.

Function name → name of member function.

Example:

```
Class test
{
private :
        int a;
        int b;
public:
        void set_data(int , int );
        int  big();

};
void  test :: set_data(int x, int y)
        {
                a=x;
                b=y;
        }
        int  test :: big()
        {
        if (a > b)
                return a;
        else
                return  b;
```

The Arrow operator

- It is also called as pointer to member access operator.
- It is used when member functions or member data has to access through a pointer which is pointing to an object of a class.
- Syntax for using arrow operator is :

Pointer_to_object -> class_member;

Example of arrow operator

```
Class test1
{
private :
    int a;
    int b;
public:
    void add(int , int );

};
void  test1 :: add(int
    x, int y)
    {
        a=x;
        b=y;
        return (a+b);
    }
```

```
void main()
{
int sum;
test1 t;
sum = t.add(4,9)
cout<< sum << endl;
test1 *t1 = &t;
sum = t1 -> add(2,7);
cout<< sum<<endl;
}
```

Output:

13

9

This operator

- It is a keyword used to store the address of the object that invokes a member function.
- When each member function is invoked this pointer implicitly holds the address of the object itself.
- It is defined internally.
- When an object is used to invoked a class member function then the address of that object is automatically assigned to the *this* pointer.

Example showing the explicit use of *this* pointer :

```
#include<iostream>
class simple
{
        int a;
    public:
        void set_data( int x)
        {
            this -> a = x;
        }
        void display()
        {
        cout<<this -> a<<endl;
        cout<<“address of the object is =”<<
    this<<endl;
        }
};
```

```
void main()
{
simple s;
s.set_data(7);
s.display();
}
```

Output:
7
address of the object
is = 0X8feeff4