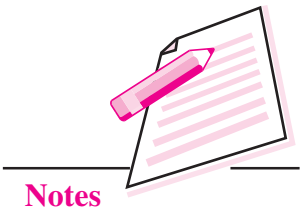


Notes

18

CLASSES AND OBJECTS WITH CONSTRUCTORS/ DESTRUCTORS

In the previous lesson you have learnt about structure, typedef and enumerated data types. In object oriented programming, the emphasis is on data rather than function. Class is a way that binds the data & function together. Constructor is a specially designed class and destructor returns the memory addresses back to the system. In this lesson you will learn about classes, objects with constructors and destructors.

OBJECTIVES

After reading this lesson, you will be able to:

- define class and object;
- access the members of the class;
- learn about three visibility modes: public, private and protected;
- define constructor with default arguments;
- use destructor.

18.1 CLASS

A class is a way to bind the data and its associated functions together. It allows data functions to be hidden, if necessary from external use. A class specification has two parts.

(i) Class declaration

(ii) Class function definitions

The general format of a class declaration is

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

The keyword class is followed by the name of the class. The body of the class is enclosed between braces and terminated by semi-colon. The class body contains the declaration of variables and functions. These are collectively called **members**. The variables declared inside the class are called as data members. The functions are known as **member functions**. The keywords public, private and protected are called as **visibility modes / access specifiers**.

The data member and member functions present in private and protected mode can be accessed by the member function in public mode. The private and protected modes are exactly the same except that protected can be inherited (explained later in this lesson) but private mode cannot be inherited. The data member and member functions present in public mode can be accessed within the class and also by other class of same program.

By default, the members of a class are private. Private data members and private functions can be accessed only by member functions of a class. Public members can be accessed from outside of the class.

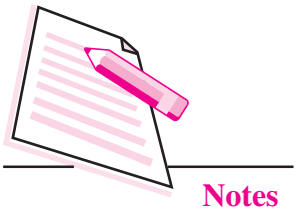
The use of keyword private is optional. By default, the members of a class are private. If the labels are missing, members are private by default.

Class Example;

```
                Class Employee              // class name Employee
                { int emp_no;               // variable … by default private
Body of class   public:
                void get_emp_no( );         // function declaration
                void put_emp_no ( );        // function declaration
                }
```

Example 1 Program to get and display Student data

```
# include < iostream.h >
class student
{
private :
    char name [ 80 ];
    int rn ;
    float marks ;
private :
    void getdata ( ) ;
    void putdata ( ) ;
};
void student : : getdata ( )
{
    cin >> name >>rn >> marks ;
}
void student : : putdata ( )
{
    cout << name << rn << marks ;
}
void main ( )
{
    student st ;
    st.getdata ( ) ;
    st.putdata ( ) ;
}
```



Notes

18.1.1 Creating Objects

An object is an instance of a class and it can be created by using class name.

In example 1 we have created object as

student st ;

Object can be created when a class is defined by placing their names immediately after the closing brace.

```
class student
{
} x, y, z ;
```

The above definition would create the object x, y and z of type student.

Accessing class member

Through object, data member and member function present in public can be accessed. The general format is :

Object name . data member ;

Object name . member function ;

The dot operator is called the **class member access operator**.

In example 1 we have accessed class member as st.getdata( ) and st.putdata( ).

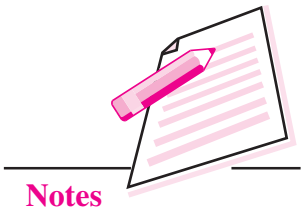
Defining member function

Member function can be defined in two ways (i) inside the class, (ii) outside the class.

(i) Inside the class: When a member function is defined inside a class, it is considered to be **inline** by default. If a member function is very small then it should be defined inside the class.

The class declaration of example 1 can be as follows:

```
class student
{
    char name [ 20 ];
    int rn ;
    float marks ;
    public :
    void getdata ( )
    {
            cin >> name >> m >> marks ;
```




```
    }
    void putdata ( )
    {
            cout << name << rn << marks;
    }
};
```

(ii) Outside the class: When a function has larger code then it should be defined outside the class declaration. The prototype of such functions, however, must be provided in the class definition. The operator ': :' is known as scope resolution operator and is used to associate member functions to their corresponding class.

The format is :

return _type class_name : : function_name

18.1.2 Nesting of member functions

A member function can be called by using its name inside another member function of the same class. This is known as **nesting of member functions**. The following program illustrates this concept.

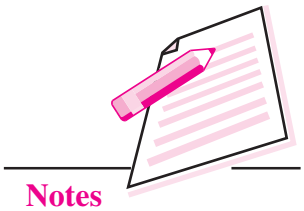
```
class greatest                          //class name
{
    int x, y, z ;                       //by default public
    public :
    void getdata ( ) ;                  //function declaration
    void display ( ) ;
    int largest ( ) ;
};
int greatest : : largest ( )            //function definition outside class
{
    int T ;
    if ( x > y )
            T = x ;
```

Notes


```
    else
                T = y ;
    if ( T > z )
                return ( T ) ;
    else
                return ( z ) ;
}
void greatest : : getdata ( )          //function definition outside class
{
    cout <<"Enter values of x, y, z" << "\n";
    cin >> x >> y >> z ;
}
void greatest : : display ( )          //function definition outside class
{
    cout << "largest value" << largest ( ) << "\n";  //nesting of member function
}
void main ( )
{
    greatest A;                        //object of class
    A. getdata ( );              //accessing class members with dot operator
    A. display ( );
}
```

18.1.3 Memory Allocation for objects

The member function are created and placed in the memory space only once when they are defined in class specification. All the objects belonging to that class use the same member functions. Space for member variable is allocated separately for each object because member variable holds different value for different objects.



Array of object

Consider the following program.

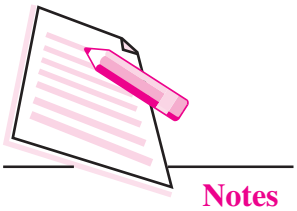
```
class emp
{
    char name [ 30 ] ;
    int empno ;
    public :
    void getdata ( ) ;
    void putdata ( ) ;
};
void main ( )
{
    emp e [ 10 ] ;
    for ( i = 0; i < 10; i ++)
    e [ i ]. getdata ( ) ;
    for (i = 0; i < 10; i ++ )
    e [ i ].putdata ( );
}
```

In the above program, the ten objects are created, namely, e [0], e [1], ... e [9].

The statement e [i]. getdata ( ) will get the data of the i^{th} element of the array e.

INTEXT QUESTIONS 18.1

1. Fill in the blanks:
   (a) All members of a class are ..... by default.
   (b) ..... is an instance of a class.
   (c) To define member function outside the class ..... operator is used.



Notes

(d) Operator : : is known as ..... operator.

(e) The variables declared inside the class are called as .....

2. State whether the following statements are true or false:

(a) A class contains only private data members.

(b) A class member is accessed using dot operator.

(c) The class declaration must end with a semicolon.

(d) Scope resolution operator is always used for defining the member functions outside the class declaration.

18.2 CONSTRUCTOR

A constructor is a special member function that initializes the objects of its class automatically when it is created. It is special because its name is the same as the class name. It is invoked automatically whenever an object is created. It is called constructor because it constructs the values of data members of the class. It does not have any return data type, not even void.

A constructor is declared and defined as follows:

```
class student
{
int rn;
int total ;
public :
    student ( )                  // constructor
    {
        rn = 0 ; total = 0 ;
    }
} ;
```

Constructor should be declared in the public section. It does not have any return data type hence it can not return any values.

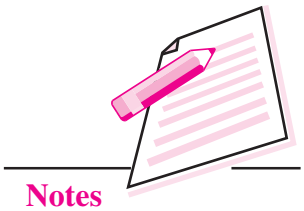
The declaration

```
student st ;
```

Invokes the constructor, student ( ) and assigns the value 0 to rn and total variables.

18.2.1 Default Constructor

A constructor that accepts no parameter is called default constructor. If no such constructor is defined, then the compiler supplies a default constructor. In that case, it is called **nothing-to-do constructor**.



Notes

18.2.2 Parameterized Constructors

The constructors that can take arguments are called parameterized constructors.

```
class student
{
int rn, total ;
public:
    student (int x, int y)              // Parameterized constructor
    {
    rn = x ; total = y ;
    }
} ;
```

When the object is created, we must supply arguments to the constructor function. This can be done in two ways:

- By calling the function explicitly
- By calling the function implicitly

The first call is implemented as follows :

```
student S1 = student ( 1, 70 ) ;
```

The second call is implemented as follows :

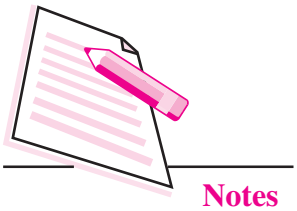
```
student S1 ( 1, 70 ) ;
```

The second method is used very often as it is shorter.

18.2.3 Copy Constructor

A copy constructor takes a reference to an object of the same class as itself as an argument. Consider the following program segment:

```
class student
{
int rn, total ;
public :
    student (int x, int y )             // Parameterized constructor
    {
        rn = x ; total = y ;
    }
```



Notes

```
    student (Student &i )              // copy constructor
    {
        rn = i. rn ;
        total = i. total ;
    }
};
```

The above program has both parameterized and copy constructor. The statement

```
student S1 ( 1, 75 ) ;
```

calls the parameterized constructor and assigns 1 to rn and 75 to total. The statement

```
student S2 (S1) ;
```

uses copy constructor and initializes an object S2 from another object S1. Another form of the statement is

```
student S2 = S1 ;
```

The process of initialization through a copy constructor is known as copy initialization.

Note that the statement

```
S2 = S1 ;
```

does not invoke the copy constructor. However, it simply assigns the value of S1 to S2, member by member.

18.2.4 Constructor with default arguments

The constructor can be declared with default arguments.

For example :

```
student ( int rn, int total = 0 ) ;
```

Here the default value of total is zero.

Then the statement

```
student S1 ( 2 ) ;
```

assigns the value 2 to rn and 0 to total.

However, the statement

```
student S2 ( 3, 75 ) ;
```

assigns 3 to rn and 75 to total. In this case actual parameter takes the priority over default parameter. All default values should be on the right side.

Consider the following statement

student (int = 0) ;

It has only one argument. It can be called in two ways.

student si( ) ;

student si(5) ;

In the first statement, no parameter is supplied. In the second statement, one parameter is supplied. When no parameter is supplied, it becomes a default constructor. When both the forms are used in a class (default constructor and constructor with one default argument), it causes ambiguity for a statement such as

student si;

(whether to call student ( ) or student (int = 0).

18.3 DESTRUCTOR

It is used to destroy the objects that have been created by a constructor. The destructor is a member function whose name is the same as the class name but is preceded by a tilde. For example the destructor of the class student can be defined as

~student ( ) ;

It never takes any argument nor does it return any value. It will be invoked by the compiler upon exit from the program (or function or block) to clean the storage. It is a good practice to declare destructor in a program because it releases memory space for future use.

INTEXT QUESTIONS 18.2

1. Fill in the blanks:

   (a) A constructor name is the same as .....

   (b) A constructor is executed automatically when an object is .....

   (c) Destructor is executed ..... at the end of the program.

   (d) A class can have any number of constructors but only ..... destructor at a time.

   (e) The name of the destructor function is same as that of class but preceded with a symbol .....

   (f) The constructor which accepts value from outside is called ..... constructor.

2. State whether the following statements are true or false:
   (a) A constructor name is not the same as class name.
   (b) In a class, you can have more than one constructor with the same name.
   (c) Constructor does not return any value.
   (d) A constructor that accepts no parameter is known as default constructor.
   (e) Destructor never takes any arguments.
   (f) Destructor is called automatically at the end of compound statement, function or main program.
   (g) Constructor initializes the data members of a class.

WHAT YOU HAVE LEARNT

- Class is a way to bind the data and its associated functions together.
- Object is an instance of a class.
- Member function can be defined inside the class and outside the class.
- Member function can be called by using its name inside another member function of the same class. This is known as nesting of member functions.
- Constructor is a special member function that initializes objects of its class. It is special because its name is the same as the class name.
- Constructor that accepts no parameter is called default constructor.
- Constructors that take arguments are called parameterized constructors.
- Destructor is used to destroy the objects that have been created by a constructor.

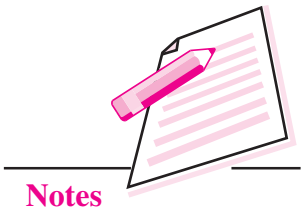
TERMINAL EXERCISE

1 What do you understand by visibility modes in class derivations ? What are these modes?

2 Define a class Teacher with the following specifications :

Private members :

Name		20 characters
Subject		10 characters
Basic, DA,	HRA	float
Salary		float



Notes

Calculate ( ) function computes the salary and returns it. Salary is sum of Basic, DA and HRA

Public members :

Readdata ( ) function accepts the data values and invokes the calculate function.

Displaydata ( ) function prints the data on the screen.

3. Define a class worker with the following specifications :

Private members of class worker

wno	integer
wname	25 characters
hrwk, wgrate	float (hour worked and wage rate per hour)
totwage	float (hrwk * wgrate)
calcwg ( )	A function to find hrwk * wgrate with float return type.

Public members of class workder

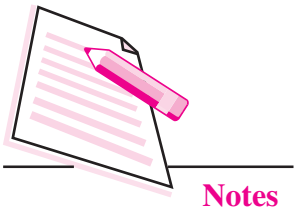
In_data ( )	a function to accept values for wno, wname, hrwk, wgrate and invoke calcwg ( ) to calculate netpay.
Out_data ( )	a function to display all the data members on the screen

4. What are the special properties of a constructor function?
5. What is parameterized constructor?
6. What is copy constructor ?
7. What is the importance of destructors ?

ANSWERS TO INTEXT QUESTIONS

18.1

1. a) private
   b) object
   c) : :
   d) Scope resolution
   e) data members.



Notes

2. a) False
   b) True
   c) True
   d) True

18.2

1 a) class name
   b) created
   c) automatically
   d) one
   e) tilde (~)
   f) parameterized

2 a) False
   b) True
   c) True
   d) True
   e) True
   f) True
   g) True