

13

Classes & Objects with Constructors/Destructors

13.1 Introduction

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.

13.2 Objectives

After going through this lesson, you would be able to

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

13.3 Class

A class is 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 data members. The functions are known as member functions. The keywords `public`, `private` and `protected` are called visibility modes.

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 (covered later) but `private` mode cannot be inherited.

Therefore, for the class concept, `private` mode is used.

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. Such a class is completely hidden from outside world and does not serve any function.

Example 1

```
# 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 ( ) ;
}
```

13.3.1 Creating Objects

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

```
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.

13.3.2 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.

13.3.3 Member function definition

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 previous program will be as follow:

```
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 ‘:.’ known as scope resolution operator is used to associate member functions to their corresponding class.

The format is :

```
return _type class_name :: function_name
```

13.3.4 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 geatest  
{  
    int x, y, z ;  
    public :  
        void getdata ( ) ;  
        void display ( ) ;  
        int largest ( ) ;  
};  
int geatest :: largest ( )  
{ int T ;  
    if ( x > y )  
        T = x ;  
    else  
        T = y ;  
    if ( T > z )  
        return ( T ) ;  
    else  
        return ( z ) ;  
}
```

```
void greatest : : getdata ( )
{
    cout << "Enter values of x, y, z" << "\n";
    cin >> x >> y >> z ;
}
void greatest : : display ( )
{
    cout << "largest value" << largest ( ) << "\n";
}
void main ( )
{
    greatest A;
    A. getdata ( ) ;
    A. display ( ) ;
}
```

13.3.5 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.

13.3.6 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 ith element of the array e.

In-Text Questions 13.1

1. What is a class ?
 2. Define object.
 3. What are the three visibility modes?
 4. What is the default mode to the members of a class?
 5. How are public members of a class different from private members of a class?
 6. Write a class specifier that creates a class called student with one private data member called rn, of type int, and two public member function : putdata () and getdata ().
 7. Fill in the blanks:
 - (a) All members of a class are by default.
 - (b) A function declared and defined inside the class definition by default is
 - (c) To define member function outside the class operator is used.
 - (d) Operator :: is known as operator.
 8. Whether the following 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 always be used for definition of member functions outside the class declaration.
-

13.4 Constructor

A constructor is a special member function that initializes the objects of its class. 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 type, not even void.

A constructor is declared and defined as follows :

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

The declaration

```
student st ;
```

Invokes the constructor, student () and assign rn = 0 and total = 0.

13.4.1 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.

13.4.2 Parameterized Constructors

The constructors that can take arguments are called parameterized constructors.

```
class student
{
    int rn, total ;
    public
    student (int x, int y)
    {
        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.

13.4.3 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 )
    {
        rn = x ; total = y ;
    }
    student (Student & i )
    {
```

```
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.

13.5 Constructor with default arguments

The constructor can be declared with default argument.

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

```
A (int = 0) ;
```

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

```
A B ;  
A B ( 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

```
A B
```

(whether to call A () or A (int = 0)

13.6 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.

In-Text Questions 13.2

1. What is a constructor?
 2. What is a default constructor?
 3. If l1 and l2 are the objects of the same class, what does the following statement do ?
l1 = l2;
-

4. A constructor is executed automatically when the object is created True or False?
 5. What is a destructor?
 6. 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 proceed with a symbol.....
 - (f) The constructor which accepts value from outside is called constructor.
 7. State whether the following 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 initialize the data members of a class.
-

13.7 What you have learnt

In this lesson, you learnt about class, object, constructor and its type and destructor. Now you should be in a position to write a program using destructor. Also, you learnt why destructor is included in a program.

13.8 Terminal Questions

1. How does a class accomplish data hiding ?
2. How does C++ structure differ from C++ class ?
3. What do you understand by visibility modes in class derivations ? What are these modes.
4. Define a class Teacher with the following specifications :

Private members :

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

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.

5. 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

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

13.9 Feedback to In-Text Questions

In-text Questions 13.1

1. Class is a way to bind data and function together.
 2. The instance (variable) of the class is called the object.
 3. public, private, protected
 4. private
 5. A public member of the class can be accessed by the object whereas private members can be accessed by the member function of public.
 6. class student
{
 int rn ;
 public :
 void getdata () ;
 void putdata () ;
};
 7. a) private
 b) inline
 c) ::
 d) Scope resolution
-

- 8. a) F
- b) T
- c) T
- d) T

In-text Questions 13.2

- 1. A special function whose name is same as the class name is called constructor. It does not have any return type. It is executed automatically when an object is created.
 - 2. A constructor with no parameter is called default constructor.
 - 3. It assigns all the members of object l2 to l1 one by one.
 - 4. True
 - 5. It is a function whose name is same as the class name but preceded by the tilde sign. It is executed automatically at the end of block or function. It releases the memory which can be used to store some other data or instruction.
 - 6.
 - a) class name
 - b) created
 - c) automatically
 - d) one
 - e) tilde (~)
 - f) parameterized
 - 7.
 - a) F
 - b) T
 - c) T
 - d) T
 - e) T
 - f) T
 - g) T
-