CHAPTER 8

CONSTRUCTORS AND DESTRUCTORS

8.1 Introduction

When an instance of a class comes into scope, a special function called the constructor gets executed. The constructor function initializes the class object. When a class object goes out of scope, a special function called the destructor gets executed. The constructor function name and the destructor have the same name as the class tag. Both the functions return nothing. They are not associated with any data type.

8.2 Constructors

```
// Program - 8.1
// to determine constructors and destructors

#include<iostream.h>
#include<conio.h>
class simple
{
 private:
        int a,b;
 public:
 simple()
 {
        a= 0 ;
        b= 0;
        cout<< "\n Constructor of class-simple ";
 }

        ~simple()
  {
        cout<<"\n Destructor of class - simple .. ";
  }

        void getdata()
 {
        cout<<"\n Enter values for a and b... ";
        cin>>a>>b;
 }

    void putdata()
    {
     cout<<"\nThe two integers .. "<<a<<'\t'<< b;
     cout<<"\n The sum of the variables .. "<< a+b;
    }
};

 void main()
{
 simple s;
 s.getdata();
 s.putdata();
}
```

When the above program is executed, constructor simple() is automatically executed when the object is created. Destructor ~ simple() is executed, when the scope of the object 's' is lost, i.e., at the time of program termination.

The output of the program will be as follows:

```
Constructor of class - simple ..
Enter values for a & b… 5 6
The two integers….. 5  6
The sum of the variables….. 11
Destructor of class - simple …
```

8.3 Functions of constructor

1) The constructor function initializes the class object

2) The memory space is allocated to an object

8.4 Constructor overloading

Function overloading can be applied for constructors, as constructors are special functions of classes. Program - 8.2 demonstrates constructor overloading.

The constructor add() is a constructor without parameters(non parameterized). It is called as default constructor. More traditionally default constructors are referred to compiler generated constructors i.e., constructors defined by the computers in the absence of user defined constructor. A non- parameterized constructor is executed when an object without parameters is declared.

```
// Program - 8.2
// To demonstrate constructor overloading

# include<iostream.h>
#include<conio.h>
class add
{
       int num1, num2, sum;
       public:
       add()
       {
              cout<<"\n Constructor without parameters.. ";
              num1= 0;
              num2= 0;
              sum = 0;
       }

       add ( int s1, int s2 )
       {
              cout<<"\n Parameterized constructor... ";
              num1= s1;
              num2=s2;
              sum=NULL;
       }

       add (add &a)
       {
              cout<<"\n Copy Constructor ... ";
              num1= a.num1;
              num2=a.num2;
              sum = NULL;
       }

       void getdata()
       {
              cout<<"Enter data ... ";
              cin>>num1>>num2;
       }
       void addition()
       {
              sum=num1+num2;
       }

       void putdata()
       {
              cout<<"\n The numbers are..";
              cout<<num1<<'\t'<<num2;
              cout<<"\n The sum of the numbers are.. "<< sum;
       }
};

void main()
{
       add a, b (10, 20) , c(b);
       a.getdata();
       a.addition();
       b.addition();
       c.addition();
       cout<<"\n Object a :  ";
       a.putdata();
       cout<<"\n Object b :  ";
       b.putdata();
       cout<<"\n Object c.. ";
       c.putdata();
}
```

OUTPUT:

Constructor without parameters….

Parameterized Constructor...

Copy Constructors…

Enter data .. 5 6

Object a:

The numbers are 5 6

The sum of the numbers are ….. 11

Object b:

The numbers are 10 20

The sum of the numbers are … 30

Object c:

The numbers are 10 20

The sum of the numbers are ….. 30

The constructor add (int s1, int s2) is called as parameterized constructor .To invoke this constructor , the object should be declared with two integer constants or variables .

Note: char, float double parameters can be matched with int data type due to implicit type conversions

For example: add a (10, 60) / add a (ivar, ivar)

The constructor add (add &a) is called as copy constructor. A copy constructor is executed:

1) When an object is passed as a parameter to any of the member functions
   Example void add::putdata(add x);

2) When a member function returns an object
   For example, add getdata();

3) When an object is passed by reference to constructor
   For example, add a; b(a);

The following program – 2 demonstrates as to when a copy constructor is executed?

```
// Program – 8.3
//  To demonstrate constructor overloading

#  include<iostream.h>
#include<conio.h>
class  add
{
        int  num1,  num2,  sum;
        public:
        add()
        {
                cout<<"\n  Constructor  without  parameters..  ";
                num1=  '\0';
                num2=  '\0';
                sum  =  '\0';
        }

        add  (  int  s1,  int  s2  )
        {
                cout<<"\n  Parameterized  constructor...  ";
                num1=  s1;
                num2=s2;
                sum=NULL;
        }

        add  (add  &a)
        {
                cout<<"\n  Copy  Constructor  ...  ";
                num1=  a.num1;
                num2=a.num2;
                sum  =  NULL;
        }

        void  getdata()
        {
                cout<<"Enter  data  ...  ";
                cin>>num1>>num2;
        }
        void  addition(add  b)
        {
                sum=num1+  num2  +b.num1  +  b.num2;
        }

        add  addition()
        {
            add a(5,6);
            sum  =  num1  +  num2  +a.num1  +a.num2;
        }

        void  putdata()
        {
                cout<<"\n  The  numbers  are..";
                cout<<num1<<'\t'<<num2;
                cout<<"\n  The  sum  of  the  numbers  are..  "<<  sum;
        }
};
```

```
void main()
{
      clrscr();
      add a, b (10, 20) , c(b);
      a.getdata();
      a.addition(b);
      b = c.addition();
      c.addition();
      cout<<"\n Object a :  ";
      a.putdata();
      cout<<"\n Object b :  ";
      b.putdata();
      cout<<"\n Object c.. ";
      c.putdata();
}
```

```
ouput of the above program
Constructor without parameters..
 Parameterized constructor...
 Copy Constructor ... Enter data ... 2  3

 Copy Constructor ...
 Parameterized constructor...
 Parameterized constructor...
 Object a :
 The numbers are..2     3
 The sum of the numbers are.. 35
 Object b :
 The numbers are..0     1494
 The sum of the numbers are.. 0
 Object c..
 The numbers are..10    20
The sum of the numbers are.. 41
```

Have you noticed as to how many times copy constructor is executed?

Copy constructor is for the following statements of Program – 8.3.

```
add c(b);// object b is passed as argument

a.addition(b); // object b is passed as argument with the member
function addition

b = c.addition(); // the member function addition is returning an
object.
```

In the above example the function addition is also overloaded. So, primarily functions declared anywhere within the program can be overloaded.

8.5 Rules for constructor definition and usage

1) The name of the constructor must be same as that of the class

2) A constructor can have parameter list

3) The constructor function can be overloaded

4) The compiler generates a constructor, in the absence of a user defined constructor

5) The constructor is executed automatically

8.6 Destructors

A destructor is a function that removes the memory of an object which was allocated by the constructor at the time of creating a object. It carries the same name as the class tag, but with a tilde (~) as prefix.

Example :

```
class simple
    {
        ____
        ____
        public :
        ~simple()
        {
       .....
        }
    }
```

8.7 Rules for destructor definition and usage

1) The destructor has the same name as that of the class prefixed by the tilde character '~'.

2) The destructor cannot have arguments

3) It has no return type

4) Destructors cannot be overloaded i.e., there can be only one destructor in a class

5) In the absence of user defined destructor, it is generated by the compiler

6) The destructor is executed automatically when the control reaches the end of class scope

Exercises

I Complete the following table

	Constructor	Destructor
1. Should be declared under	- scope	- scope
2. overloading is	-	-
3. Is executed when an object is		
The function of a		

II. Why do the following snippets throw error ?

```
Class simple
{
  private :
   int x;
  simple()
 { x = 5; }
};
```

```
Class simple
{
  private :
   int x;
 public :
  simple(int y)
 { x = y; }
};
void main()
{
  simple s;
}
```

```
Class simple
{
  private :
   int x;
 public :
  simple(int y)
 { x = y; }

 simple(int z = 5)
{
    x = z;
}
};
void main()
{
  simple s(6);
}
```