CS-112
Object Oriented Programming
(3+1)
Prerequisites:
Programming Fundamentals

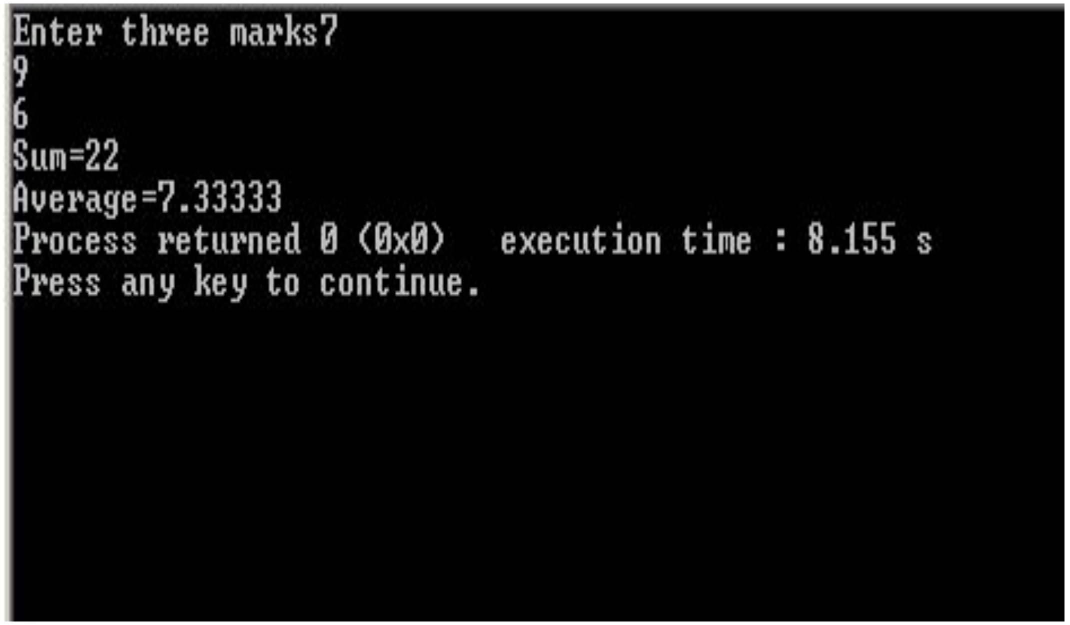
Syed Muhammad Rafi
Lecturer, Department of Software Engineering
Faculty of Engineering Science and Technology,
Ziauddin University

Constructor and Destructor

Lec-5

- Write a class Marks with three data members to store three marks.
  Write three members functions in() to input marks, sum() to calculate and return the sum and avg() to calculate and return the sum and avg() to calculate and return the average marks.

```
#include <iostream>
using namespace std;
class Marks
{
private:
int a,b,c;
public:
void in()
   {
      cout<<"Enter three marks";
      cin>>a>>b>>c;
   }
int sum()
   {
      return a+b+c;
   }
 float avg()
   {
      return(a+b+c)/3.0;
   };

int main()
{
   Marks m;
   int s;
   float a;
   m.in();
   s=m.sum();
   a=m.avg();
   cout<<"Sum="<<s<<endl;
   cout<<"Average="<<a;
   return 0;
}
```



```
Enter three marks7
9
6
Sum=22
Average=7.33333
Process returned 0 (0x0)   execution time : 8.155 s
Press any key to continue.
```

Defining Member Functions outside the Class

- The member functions of a class can also be defined outside the class.
- The declaration of member functions is specified outside class.
- The scope **resolution operator ::** is used as a function declarator if function is defined outside the class.

Syntax

The syntax of defining member function the class is as follows:

```
 return_type class_name:: function _name (parameters)
{
Function body
}
```

Where,

return_type	It indicates the type of value to be returned by the function.
Class_name	It indicates the name of the class to which the function belongs.
::	It is the scope resolution operator to define member function outside the class.
function_name	It is the name of function to be defined.

- ❑ Write a class Array that contains an array of integer to store five values.

It also contains the following member functions

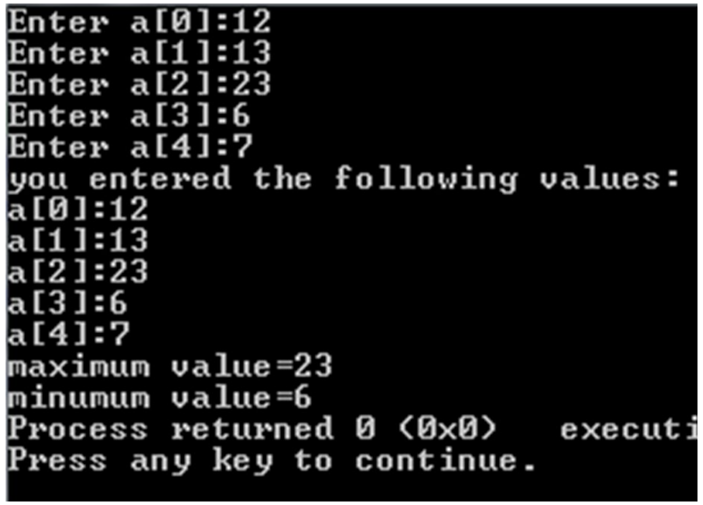
- The **fill()** function is used to fill the array with the values from the user.
- The **display()** function is used to display the values of the array .
- The **max()** function shows the maximum values in the array.
- The **min()** function shows the minimum value in the array.

All member functions should be defined outside the class.

```
# include <iostream>
using namespace std;
class Array
{
private:
int a[5];
public:
void fill();
void display();
int max();
int min();
};
void Array::fill()
{
int i;
for(i=0;i<5;i++)
{
 cout<<"Enter a["<<i<<"]:";
 cin>>a[i];
}
}
```

```
void Array::display()
{
 int i;
 for(i=0;i<5;i++)
cout<<"a["<<i<<"]:"<<a[i]<<endl;
}
int Array::max()
{
 int m=a[0];
 for(int i=0;i<5;i++)
 if(m<a[i])
 m=a[i];
 return m;
}
```

```
int Array::min()
{
 int m=a[0];
 for(int i=0;i<5;i++)
 if(m>a[i])
 m=a[i];
 return m;
}
int main()
{
   Array arr;
   arr.fill();
   cout<<"you entered the following values:"<<endl;
   arr.display();
   cout<<"maximum value="<<arr.max()<<endl;
   cout<<"minumum value="<<arr.min();
 }
```



```
Enter a[0]:12
Enter a[1]:13
Enter a[2]:23
Enter a[3]:6
Enter a[4]:7
you entered the following values:
a[0]:12
a[1]:13
a[2]:23
a[3]:6
a[4]:7
maximum value=23
minumum value=6
Process returned 0 (0x0)   executi
Press any key to continue.
```

Constructors

- A type of member function that can work automatically when an object of that class is created is known as **constructor.**
- The constructor has no return type and has same name that of class name.
- The constructor can work as a normal function but it cannot return any value.
- It is normally defined in class to initialize data member.

Syntax

The syntax of declaring constructor is as follows

```
name()
{
Constructor body
}
```

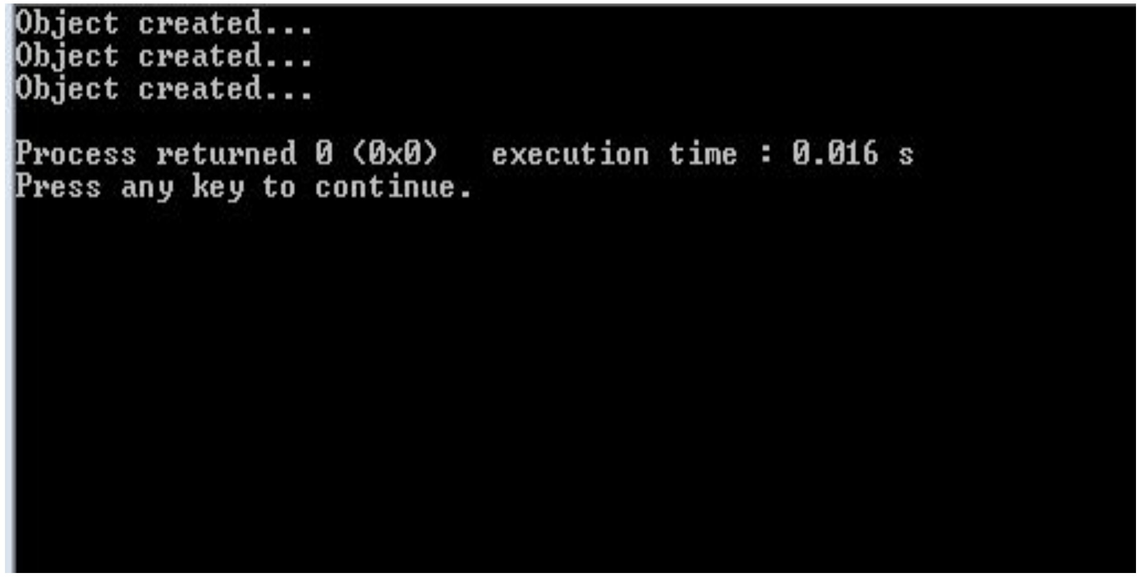
Where,

name It indicates the name of the constructor

the name must be same as the name of class in which the constructor is declared.

- ❑ Write a class that displays a simple message on the screen whenever an object of that class is created.

```
#include <iostream>
using namespace std;
class Hello
{
   private:
   int n;
   public:
   Hello()
   {
      cout<<"Object created..."<<endl;
   }

};

int main()
{
   Hello x,y,z;
   return 0;
}
```



```
Object created...
Object created...
Object created...

Process returned 0 (0x0)   execution time : 0.016 s
Press any key to continue.
```

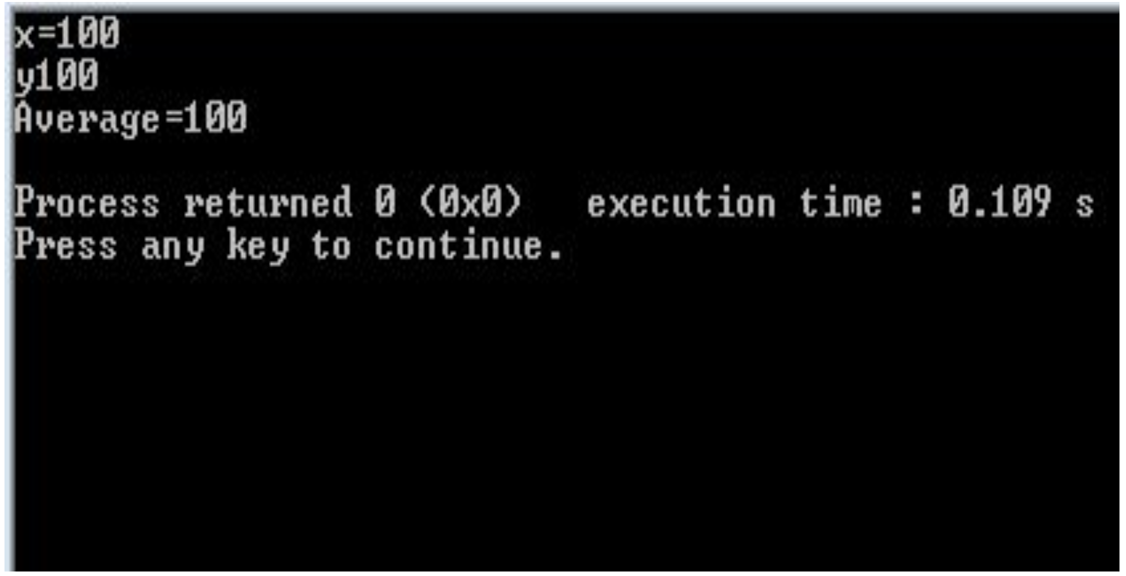
How above program works

- The above program declares a constructor that displays a message on the screen.
- The program creates three object in main() function .
- The constructor is executed each time an object of the class is created in the memory.

- Write a class that contains two integer data members which are initialized to 100 when an object is created. It has a member function **avg** that displays the average of data members.

```
#include <iostream>
using namespace std;
class Number
{
private:
int x,y;
public:
Number()
{
  x=y=100;
}
  void avg()
  {
   cout<<"x="<<x<<endl;
   cout<<"y"<<y<<endl;
   cout<<"Average="<<(x+y)/2<<endl;
  }

};
```

```
int main()
{
   Number n;
   n.avg();
   return 0;
}
```



```
x=100
y100
Average=100

Process returned 0 (0x0)   execution time : 0.109 s
Press any key to continue.
```

Passing parameters to Constructors

- The method of passing parameters to the constructor is same as passing parameters to normal functions.
- The only difference that parameters are passed to the constructor when the object is declared.
- The parameters are written parenthesis along with the object name in declaration statement.

Syntax

The syntax of passing parameters to constructor is as follows:

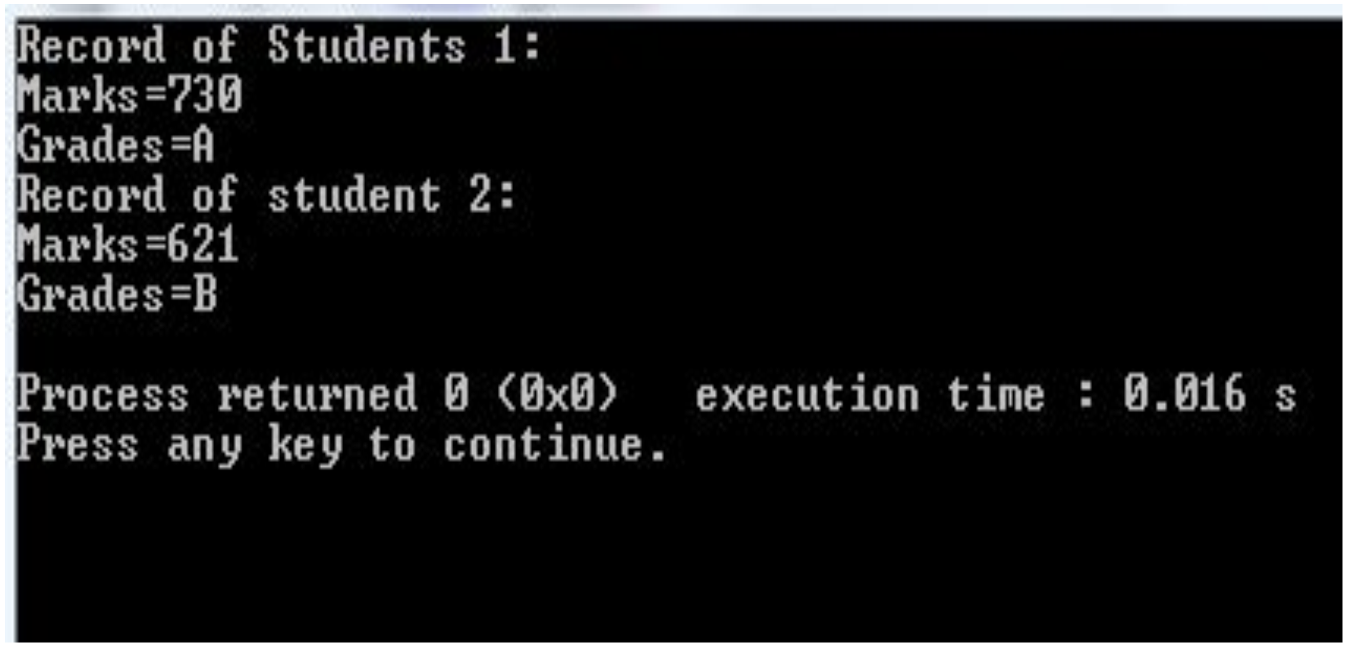
```
Type object_name(parameters);
```

Where,

type	It is the name of class and indicates the type of object to be created
Object_name	It indicates the name of object to be created.
Parameters	It indicates the list of parameters passed to constructor.

- Write a class that has marks and grade as data members. A constructor with two parameters initializes data members with the given values and member functions show displays the values of data members. Create two objects and displays the values.

```
#include <iostream>

using namespace std;
class Student
{
private:
int marks;
char grade;
public:
Student (int m, char g)
{
   marks=m;
   grade=g;
}
void show()
{
   cout<< "Marks="<<marks<<endl;
   cout<<"Grades="<<grade<<endl;
}
};

int main()
{
   Student s1(730,'A'),s2(621,'B');
   cout<<"Record of Students 1:"<<endl;
   s1.show();
   cout<<"Record of student 2:"<<endl;
   s2.show();
   return 0;
}
```



```
Record of Students 1:
Marks=730
Grades=A
Record of student 2:
Marks=621
Grades=B

Process returned 0 (0x0)   execution time : 0.016 s
Press any key to continue.
```

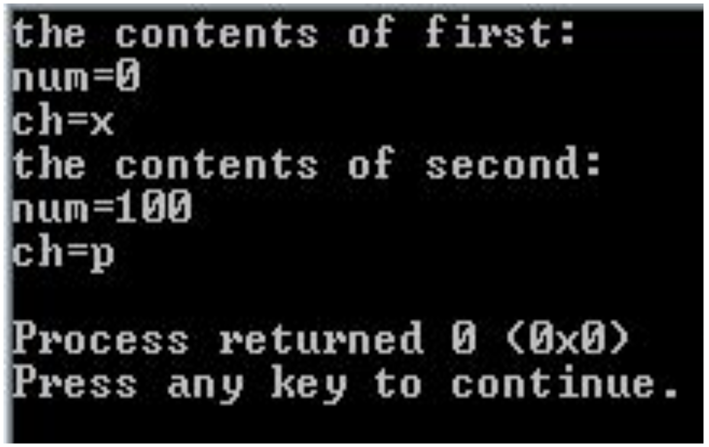
Constructor Overloading

- The process of declaring multiple constructor with same name but different parameters is known as **constructor overloading.**
- The constructor with same name must differ in one of the following ways:

1. Number of parameters
2. Types of parameters
3. Sequence of parameters

- Write a class that has **num** and **ch** as data members. A constructor with no parameter initializes num to 0 and ch to ‘x’.
  A constructor with two parameters initializes data members with the given values and member function show displays the values of data members

```
#include <iostream>
 using namespace std;
class Over
{
private:
int num;
char ch;
public:
Over()
{
num=0;
ch='x';
}
Over(int n, char c)
{
 num=n;
 ch=c;
}
void show()
{
cout<<"num="<<num<<endl;
cout<<"ch="<<ch<<endl;
}
};

int main()
{
   Over first, second(100, 'p');
   cout<<"the contents of first:"<<endl;
   first.show();
   cout<<"the contents of second:"<<endl;
   second.show();

return 0;
}
```



```
the contents of first:
num=0
ch=x
the contents of second:
num=100
ch=p

Process returned 0 (0x0)
Press any key to continue.
```

Copy constructor

- A copy constructor is a member function which initializes an object using another object of the same class.

Syntax

The syntax of copy constructor is as follows

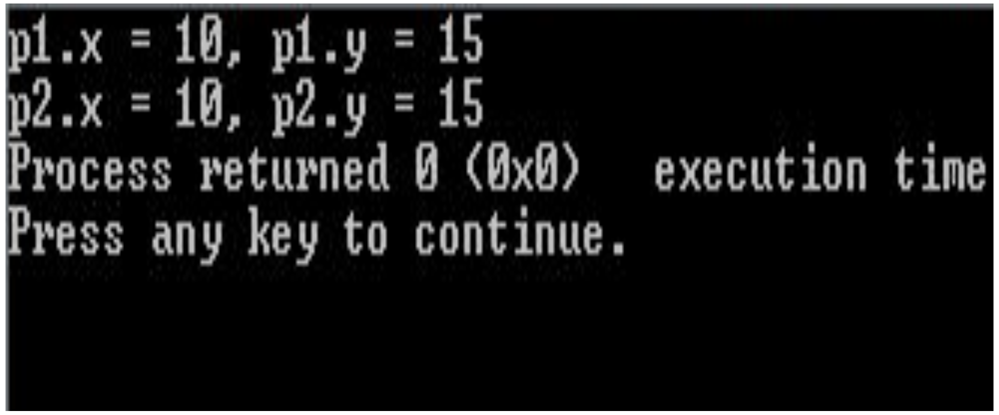
Class Name (const Class Name &Obj Name);

Where,

Class Name	It indicates the name of class and indicates the type of object to be created.
const	It is a keyword that is used for copy constructor.
&Obj Name	Obj is a reference to an object that is being used to initialize another object.

```
#include<iostream>
using namespace std;

class Point
{
private:
   int x, y;
public:
   Point(int x1, int y1)
{
x = x1;
y = y1;
}

   // Copy constructor
   Point(const Point &p2)
{
x = p2.x;
y = p2.y;
}
```

```
 int getX()
{
return x;
}
   int getY()
{
return y;
}
};

int main()
{
   Point p1(10, 15); // Normal constructor is called here
   Point p2 = p1; // Copy constructor is called here

   // Let us access values assigned by constructors
   cout << "p1.x = " << p1.getX() << ", p1.y = " << p1.getY();
   cout << "\np2.x = " << p2.getX() << ", p2.y = " << p2.getY();

   return 0;
}
```



```
p1.x = 10, p1.y = 15
p2.x = 10, p2.y = 15
Process returned 0 (0x0)   execution time
Press any key to continue.
```

Default Copy Constructor

- A type of constructor that is used to initialize an object with another object of the same type is known as **default copy constructor**.
- Its name is “default copy constructor” because it is available by default in all classes.
- The user does not need to write this constructor.
- It accepts a single object of the same type as parameter.
- The parameter for default copy constructor can be given in parenthesis or using assignment operator.

Syntax

The syntax of default copy constructor is as follows

```
class_name  object_name  (parameter);   OR
class_name  object_name  = (parameter);
```

Where,

class_name	It indicates the name of class and indicates the type of object to be created.
object_name	It indicates the name of object to be created.
Parameter	It indicates the name of parameter that is passed by default copy constructor.

Example

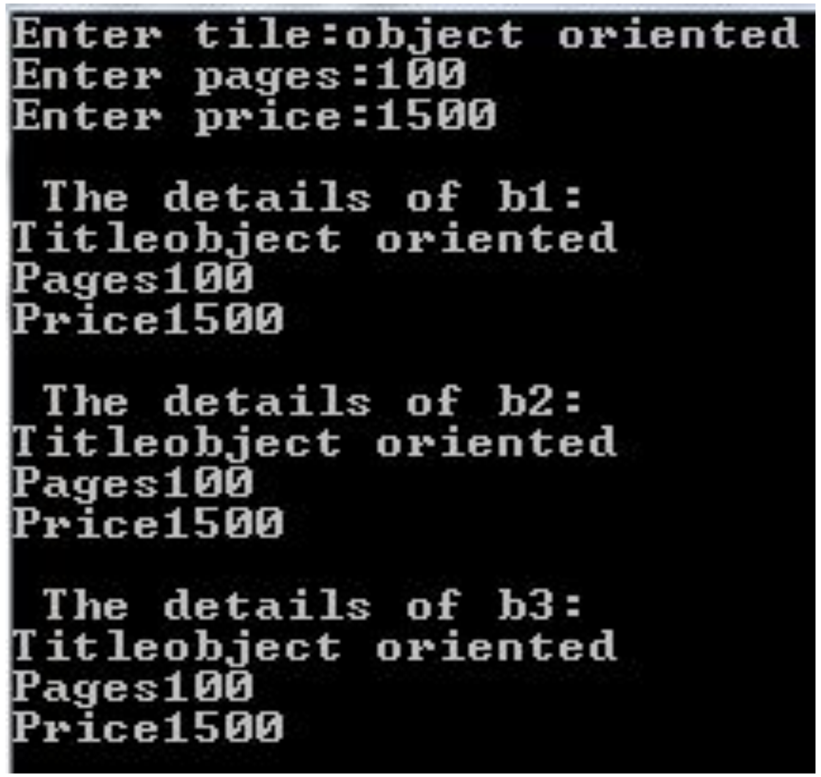
- An object can be initialized with another object of same type. This is same as copying the contents of a class to another class.
- In the above program, if you want to initialize an object A3 so that it contains same values as A2, this can be performed as:

```
....
int main()
{
  Area  A2(2, 1);

  // Copies the content of A2 to A3
  Area A3(A2);
    OR,
  Area A3 = A2;
}
```

You might think, you need to create a new constructor to perform this task. But, no additional constructor is needed. This is because the copy constructor is already built into all classes by default.

- Write a class **Book** that has attributes for pages, price and title. It has two functions to input the values and display the values. create three objects of class and input values.

```
#include <iostream>

#include <cstring>

using namespace std;


class Book

{

private:

int pg, pr;

char title[50];

public:

void get()

{

cout <<"Enter tile";

gets(title);
cout<<"Enter pages";
cin>>pg;
cout<<"Enter price";
cin>>pr;
void show()
{
cout<<"Title"<<title<<endl;
cout<<"Pages"<<pg<<endl;
cout<<"Price"<<pr<<endl;
}
};
int main()
{
Book b1;
b1.get();
Book b2(b1);
Book b3=b1;
cout<<"\n The details of b1:"<<endl;
b1.show();
cout<<"\n The details of b2:"<<endl;
b2.show();
cout<<"\n The details of b3:"<<endl;
b3.show();

return 0;
}
```



```
Enter tile:object oriented
Enter pages:100
Enter price:1500

 The details of b1:
Titleobject oriented
Pages100
Price1500

 The details of b2:
Titleobject oriented
Pages100
Price1500

 The details of b3:
Titleobject oriented
Pages100
Price1500
```

Destructor

- A type of member function that is executed when an object of that class is destroyed is known as destructor.
- Destructor is a member function which destructs or deletes an object.
- The destructor has no return type and its name is same as class name.
- The destructor cannot return any value.
- It also cannot accept any parameters.
- The constructor name is preceded by tilde sign ~.

When is destructor called?

A destructor function is called automatically when the object goes out of scope:

(1) the function ends

(2) the program ends

(3) a block containing local variables ends

(4) a delete operator is called

Syntax

The syntax of declaring destructor is as follows:

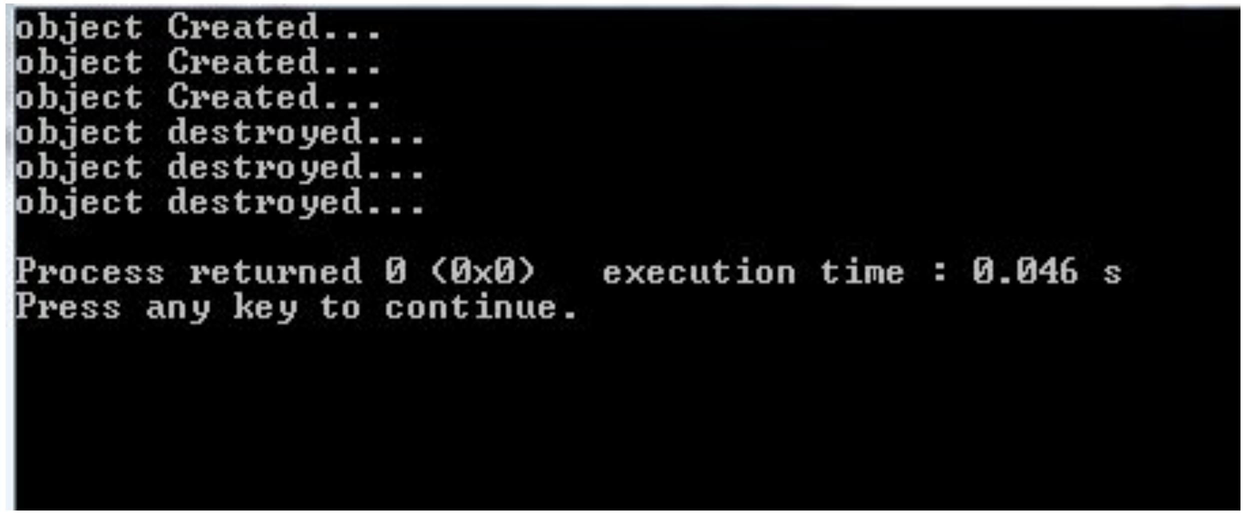
```
~name()
{
 destructor body
 }
```

Where,

~name	It indicates the name of destructor. The name must be same as that of class in which the destructor is declared.

❑ Write a program to explain the concept of destructor.

```
# include<iostream>
using namespace std;
class Test

{
private:
int n;
public:
Test()
{
 cout<<"object Created..."<<endl;
}
~Test()
{
 cout<<"object destroyed..."<<endl;
}
};
int main()
{
  Test a,b,c;
}
```



```
object Created...
object Created...
object Created...
object destroyed...
object destroyed...
object destroyed...

Process returned 0 (0x0)   execution time : 0.046 s
Press any key to continue.
```

How Above Program works

- The above program creates a constructor and destructor in the class .
- Both display simple messages on the screen.
- The message “object created ..” will appear when the program is executed.
- The message “object destroyed ..” will appear when the program is terminated and all objects are destroyed from memory.

End of lecture