Constructors & Destructors

Constructor is a special member function whose task is to initialize the objects of its class. The constructor is invoked whenever an object of its associated class is created.

```
Class Ratio
{
public:
Ratio (int n, int d) {num=n; den=d;
Void print(){ cout<<num<<'/'<<den;}
 private:
 int num,den;
};
Void main()
{Ratio x(-1,3),y(22,7);cout<<"x=";
x.print();
Cout<<"and y=";
y.print()
}
```

- Output:

x=-1/3 and y= 22/7

Constructor function has some special characteristics. These are

- Constructor function must have the same name as the class itself and it is declared without a return type.
- They should be declared in the public section.
- They cannot be inherited.
- Constructor cannot be virtual.

Multiple constructors in a class

```
Class Ratio
{ public:
Ratio() {num=0; den=1;}
Ratio (int n) {num=n; den=1}
Ratio (int n, int d){num=n; den=d;}
void print(){ cout<<num<<'/'<<den;}
 private:
 int num, den;
};
```

```
 int main()
{ Ratio x,y(4),z(22,7);
  cout<<“x=”;
  x.print();
  cout<< “\n y=”;
y.print();
Cout<<“\n z=”;
 z.print();
 }
```

Output:

x=0/1

y=4/1

z=22/7

Constructor Initialization lists

Ratio (int n, int d): num (n), den (d){ }

The assignment statements in the function's body that assigned n to num and d to den are removed.

```
Class Ratio
{ public:
   Ratio() : num(0), den(1){ }
   Ratio( int n): num( n), den(1){ }
   Ratio( int n, int d): num (n), den (d) { }
   private:
   int num, den;
}
Void main()
{ Ratio x, y(4), z(22,7)
```

x will represent 0/1

y will represent 4/1

z will represent 22/7

Copy constructors

Copy constructor is used to declare and initialize an object from another object. Copy constructor takes one parameter : the object that is going to copy . That object is passed by constant reference because it should not be changed.

```
Class Ratio
{ public:
  Ratio( int n, int d): num (n), den (d) { }
  Ratio ( count Ratio & r): num (r.num), den (r.den) { }
  void print ( ){ cout<<num<<'/'<<den;}
  private:
  int num, den;}
```

```
Void main( )
{ Ratio (5,18);
   Ratio y (x);
   cout<<“x=”;
  x.print ( );
  cout<<“y=”;
  y.print( );
}
```

Output

x=5/18 , y = 5/18

The copy constructor copies the num and den fields of the parameter r into the object being constructed. When y is declared, it calls the copy constructor which copies x into y.

The copy constructor is called automatically whenever

- An object is copied by means of a declaration initialization.
- An object is passed **by value** to a function.
- An object is returned **by value** from a function.

The class destructor

The destructor, as the name implies is used to destroy objects that have been created by a constructor . Like a constructor , the destructor is a member function whose name is the name as the class but is preceded by a tilde(~).

```
Class Ratio
{ public:
  Ratio( ){ cout<<“OBJECT IS BORN \n”;}
 ~Ratio( ){ cout<<“OBJECT DIES\n”;}
private:
int num, den;
};
```

```
 void main( )
{
  { Ratio x;
    cout<<“now x is alive.\n”;
   }
   cout<<“ now between blocks.\n”;
  { Ratio y;
    cout<< “ now y is alive. \n”;
   }
}
```

Output

OBJECT IS BORN.
Now x is alive.
OBJECT DIES.
Now between blocks.
OBJECT IS BORN
Now y is alive
OBJECT DIES.

NOTE:

new——— is used to allocate memory in the constructors.
delete—— is used to free memory in the destructors.