- Destructor is a special class function which destroys the object as soon as the scope of object ends. The destructor is called automatically by the compiler when the object goes out of scope.
- A destructor never takes any argument nor does it return any value.
- It will be called implicitly by the compiler upon exit from the program(or block or function as the case may be).

```
class Line {
public: void setLength( double len );
double getLength( void );
Line();
~Line();
private: double length;
};

Line::Line(void)
{
 cout << "Object is being created" << endl;
}
 Line::~Line(void)
{
cout << "Object is being deleted" << endl;
}
void Line::setLength( double len )
{
length = len;
}
double Line::getLength( void )
{
 return length;
}
```

```
int main()
{
 Line line;
line.setLength(6.0);
cout << "Length of line : " << line.getLength() <<endl;
return 0;
 }
```

```
class A
{
A()
{
cout << "Constructor called";
}
~A()
{
cout << "Destructor called";
}
 };
int main()
{
A obj1; // Constructor Called
int x=1
if(x)
{
A obj2; // Constructor Called
} // Destructor Called for obj2
} // Destructor called for obj1
```

Arrays with in a class

```
const int size=5;
class student
{
int roll_no;
int marks[size];
public:
void getdata ();
void tot_marks ();
} ;
void student :: getdata ()
{ cout<<"\nEnter roll no: ";
Cin>>roll_no;
for(int i=0; i<size; i++)
{ cout<<"Enter marks in subject"<<(i+1)<<": ";
cin>>marks[i] ;
}
}
```

```
void student :: tot_marks() //calculating total marks
{
int total=0;
for(int i=0; i<size; i++)
total+ = marks[i];
cout<<"\n\nTotal marks "<<total;
}
void main()
{
 student stu;
 stu.getdata() ;
 stu.tot_marks() ;
 getch();
 }
```

Reference Variable

- A **reference variable** is an alias, that is, another name for an already existing **variable**. Once a **reference** is initialized with a **variable**, either the **variable** name or the **reference** name may be used to **refer** to the **variable**.
- A **reference variable must be initialized at the time of declaration. This establishes the correspondence between the reference and the data element which it names.**

Reference Variable

```
void main()
{
int x=123;
int &y=x;
cout<<“x= “<<x<<endl;
cout<<“y= “<<y<<endl;
y++;
cout<<“x= “<<x<<endl;
getch();
}
```

- **Dynamic constructor**

Dynamic constructor is used to allocate memory to the objects at the run time. Memory is allocated at run time with the help of new operator. By using this constructor we can dynamically initialize the objects.

Dynamic constructor

```
class A()
{
int a,b;
int *p;
public:
A(int x, int y, int z)
{a=x;b=y;
p = new int;
*p =z;
}
void write()
{cout<<a<<b;
cout<<*p;}
};
 main()
{A o1(3,4,5);
o1.write()
}
```