**Destructors in C++**

A destructor is a special member function that works just opposite to constructor, unlike constructors that are used for initializing an object, destructors destroy (or delete) the object.

**Syntax of Destructor**

~class\_name()

{



//Some code

}

Similar to constructor, the destructor name should exactly match with the class name. A destructor declaration should always begin with the tilde(~) symbol as shown in the syntax above.



**When does the destructor get called?**

A destructor is **automatically called** when:  
1) The program finished execution.  
2) When a scope (the { } parenthesis) containing local variable ends.  
3) When you call the delete operator.

**Destructor Example**

#include <iostream>

using namespace std;

class HelloWorld{

public:

//Constructor

HelloWorld(){

cout<<"Constructor is called"<<endl;

}

//Destructor

~HelloWorld(){

cout<<"Destructor is called"<<endl;

}

//Member function

void display(){

cout<<"Hello World!"<<endl;

}

};

int main(){

//Object created

HelloWorld obj;

//Member function called

obj.display();

return 0;

}

**Output:**

Constructor is called

Hello World!

Destructor is called

**Destructor rules**

1) Name should begin with tilde sign(~) and must match class name.  
2) There cannot be more than one destructor in a class.  
3) Unlike constructors that can have parameters, destructors do not allow any parameter.  
4) They do not have any return type, just like constructors.  
5) When you do not specify any destructor in a class, compiler generates a default destructor and inserts it into your code.