# Destructors

# Destructor

**C++ destructor** is a special member function that is executed automatically when an object is destroyed that has been created by the constructor.

C++ destructors are used to de-allocate the memory that has been allocated for the object by the constructor.

The destructor cannot be overloaded. It is not possible to overwhelm the destructor. In a class, we can only have one destructor.

cslearnerr.com

# Destructor

- Its syntax is same as constructor except the fact that it is preceded by the tilde sign.

~class_name()

{ }; cslearnerr.com

- Unlike constructor a destructor neither takes any arguments nor does it returns value. And destructor can't be overloaded.

- **When does the destructor get called?**

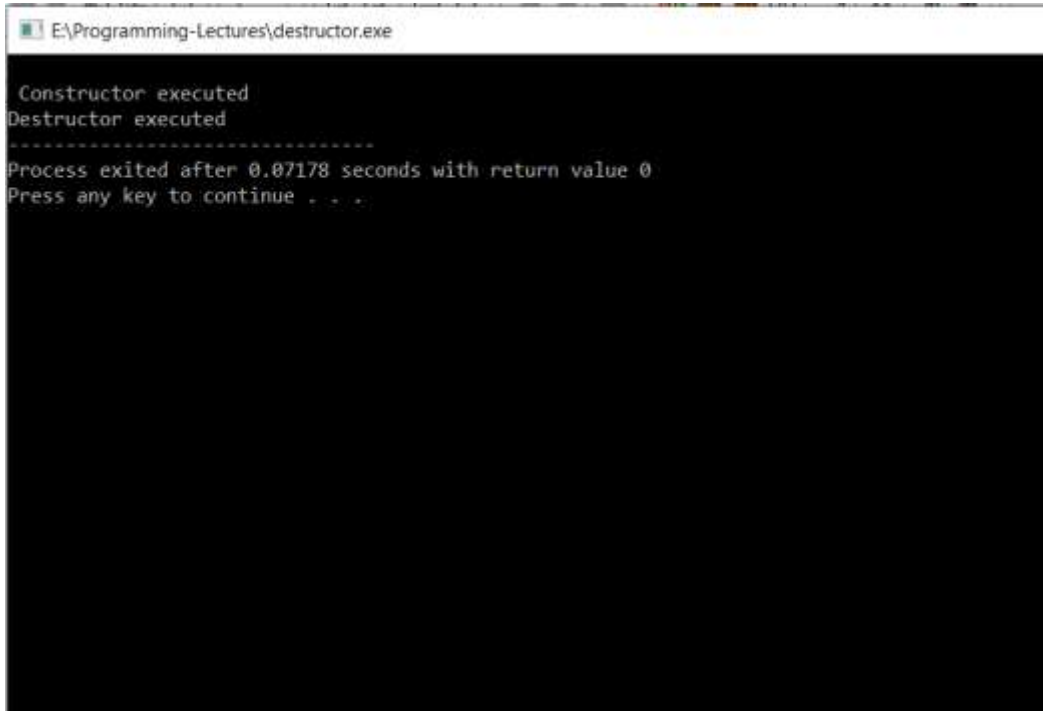
A destructor is **automatically called** when:
1) The program finished execution.
2) When a scope (the { } parenthesis) containing **local variable** ends.
3) When you call the delete operator.

# Example

destructor.cpp

```cpp
// C++ program to demonstrate the execution of constructor
// and destructor

#include <iostream>
using namespace std;

class Test {
public:
    // User-Defined Constructor
    Test() { cout << "\n Constructor executed"; }

    // User-Defined Destructor
    ~Test() { cout << "\nDestructor executed"; }
};
main()
{
    Test t;

    return 0;
}
```





# Friend Function

cslearnerr.com

A friend function in C++ is defined as a function that can access private, protected and public members of a class.

The friend function is declared using the friend keyword inside the body of the class.

**Friend Function Syntax:**

By using the keyword, the **'friend'** compiler understands that the given function is a friend function.

We declare friend function inside the body of a class, whose private and protective data needs to be accessed, starting with the keyword friend to access the data. We use them when we need to operate between two different classes at the same time.

```
1  class className {
2      ... .. ...
3      friend returnType functionName(arguments);
4      ... .. ...
5  }
```

# Example

friend func.cpp | friend new func.cpp

```
1  #include<iostream>
2  using namespace std;
3  class B;
4  class A
5  {
6  private:
7   int a;
8   public:
9      A()
10     {
11         a=10;
12     }
13     friend void show(A,B);
14 };
15 class B
16 {
17     private:
18         int b;
19         public:
20             B()
21             {
22                 b=20;
23             }
24             friend void show(A,B);
25 };
26 void show (A x, B y)
27 {
28 int sum;
29 sum=x.a+y.b;
30 cout<<"The value of class A object = "<<x.a<<endl;
31 cout<<"The value of class B object = "<<y.b<<endl;
32 cout<<"The sum of objects = "<<sum<<endl;
33 }
34 int main()
35 {
36     A obj1;
37     B obj2;
38     show(obj1,obj2);
39 return 0;
40 }
```

```
E:\Programming-Lectures\friend new func.exe
The value of class A object = 10
The value of class B object = 20
The sum of objects = 30

--------------------------------
Process exited after 0.03369 seconds with return value 0
Press any key to continue . . .
```

# Example

cslearnerr.com

friend func.cpp

```
#include<iostream>
using namespace std;
class A
{
    private:
        int a,b;
        public:
        A()
        {
            a=10;
            b=20;
        }
        friend class B;
};
class B
{
    public:
    void showA(A obj)
    {
        cout<<"The value of a:  "<<obj.a<<endl;
    }
    void showB(A obj)
    {
    cout<<"The value of b:  "<<obj.b<<endl;
    }
};
int main()
{
    A x;
    B y;
    y.showA(x);
    y.showB(x);
    return 0;
}
```

cslearnerr.com

E:\Programming-Lectures\friend func.exe

```
The value of a:  10
The value of b:  20

--------------------------------
Process exited after 0.237 seconds with return value 0
Press any key to continue . . .
```