

Base Class Constructor and Destructor

In this lesson, we'll learn how constructors and destructors are called in derived and base classes during inheritance.

WE'LL COVER THE FOLLOWING ^

- Base Class Constructor
- Base Class Destructor

Base Class Constructor

When we make an instance of the **Derived** class without parameters it will first call the default constructor of the **Base** class and then the **Derived** class. In the same way, when we call the parameterized constructor of the derived class, it will first call the parameterized constructor of the **Base** class and then **Derived** class.

The following code explains how this is done:

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

// Base class
class Base {

    public:
    Base(){
        cout << "Base class default constructor!" << endl;
    }
    // Base class's parameterised constructor
    Base(float i) {
        cout << "Base class parameterized constructor" << endl;
    }
};

// Derived class
class Derived : public Base {
    public:
    Derived(){
        cout << "Derived class default constructor!" << endl;
```



```

    }

    // Derived class's parameterised constructor
    Derived(float num): Base(num){
        cout << "Derived class parameterized constructor" << endl;
    }
};

// main function
int main() {
    // creating object of Derived Class
    Derived obj;
    cout << endl;
    Derived obj1(10.2);
}

```



Base Class Destructor

When we make an instance of the **Derived** class it will first call the destructor of the **Derived** class and then the **Base** class.

The following code explains how this is done:

```

#include <iostream>
using namespace std;

// Base class
class Base {

    public:
    ~Base(){
        cout << endl << "Base class Destructor!" ;
    }
};

// Derived class
class Derived : public Base {
    public:

    ~Derived(){
        cout << endl << "Derived class Destructor!" ;
    }
};

// main function
int main() {
    // creating object of Derived Class
    Derived obj;
}

```





In the next lesson, we'll be learning about the `public`, `protected` and `private` inheritance.