Exception Handling and Object Destruction in C++

Difficulty Level : Easy • Last Updated : 20 Jan, 2023

An exception is termed as an unwanted error that arises during the runtime of the program. The practice of separating the anomaly-causing program/code from the rest of the program/code is known as Exception Handling.

An object is termed as an instance of the class which has the same name as that of the class. A destructor is a member function of a class that has the same name as that of the class but is preceded by a **'~'** (tilde) sign, also it is automatically called after the scope of the code runs out. The practice of pulverizing or demolition of the existing object memory is termed *object destruction*.
In other words, the class of the program never holds any kind of memory or storage, it is the object which holds the memory or storage and to deallocate/destroy the memory of created object we use destructors.

For Example:

CPP

```
// C++ Program to show the sequence of calling
// Constructors and destructors
#include <iostream>
using namespace std;

// Initialization of class
class Test {
public:
    // Constructor of class
    Test()
    {
        cout << "Constructing an object of class Test "
             << endl;
    }

    // Destructor of class
    ~Test()
    {
        cout << "Destructing the object of class Test "
             << endl;
    }
};

int main()
{
    try {
        // Calling the constructor
        Test t1;
        throw 10;

    } // Destructor is being called here
      // Before the 'catch' statement
    catch (int i) {
        cout << "Caught " << i << endl;
    }
}
```

Output:

AD

```
Constructing an object of class Test
Destructing the object of class Test
Caught 10
```

When an exception is thrown, destructors of the objects (whose scope ends with the try block) are automatically called before the catch block gets executed. That is why the above program prints ***"Destructing an object of Test***" before "***Caught 10***".

What Happens When an Exception is Thrown From a Constructor?

Example:

CPP

```cpp
// C++ Program to show what really happens
// when an exception is thrown from
// a constructor
#include <iostream>
using namespace std;

class Test1 {
public:
    // Constructor of the class
    Test1()
    {
        cout << "Constructing an Object of class Test1"
             << endl;
    }
    // Destructor of the class
    ~Test1()
    {
        cout << "Destructing an Object the class Test1"
             << endl;
    }
};

class Test2 {
public:
    // Following constructor throws
    // an integer exception
    Test2() // Constructor of the class
    {
        cout << "Constructing an Object of class Test2"
             << endl;
        throw 20;
    }
    // Destructor of the class
    ~Test2()
    {
        cout << "Destructing the Object of class Test2"
             << endl;
    }
};

int main()
{
    try {
        // Constructed and destructed
        Test1 t1;

        // Partially constructed
        Test2 t2;

        // t3 is not constructed as
        // this statement never gets executed
        Test1 t3; // t3 is not called as t2 is
                  // throwing/returning 'int' argument which
```

```
                    // is not accepted
                    //  is the class test1'
        }
        catch (int i) {
            cout << "Caught " << i << endl;
        }
    }
```

Output:

```
Constructing an Object of class Test1
Constructing an Object of class Test2
Destructing an Object the class Test1
Caught 20
```

Destructors are only called for the completely constructed objects. When the constructor of an object throws an exception, the destructor for that object is not called.

Predict the output of the following program:

CPP

```
// C++ program to show how many times
// Constructors and destructors are called
#include <iostream>
using namespace std;

class Test {
    static int count; // Used static to initialise the scope
                      // Of 'count' till lifetime
    int id;

public:
  // Constructor
    Test()
    {
        count++;
        id = count;
        cout << "Constructing object number " << id << endl;
        if (id == 4)
            throw 4;
    }
  // Destructor
    ~Test()
    {
        cout << "Destructing object number " << id << endl;
    }
};

int Test::count = 0;
```

```
// Source code
int main()
{
    try {
        Test array[5];
    }
    catch (int i) {
        cout << "Caught " << i << endl;
    }
}
```

Output :

```
Constructing object number 1
Constructing object number 2
Constructing object number 3
Constructing object number 4
Destructing object number 3
Destructing object number 2
Destructing object number 1
Caught 4
```

40

Related Articles

1. Virtual destruction using shared_ptr in C++
2. Comparison of Exception Handling in C++ and Java
3. Top C++ Exception Handling Interview Questions and Answers
4. Exception Handling in C++
5. C++ | Exception Handling | Question 1
6. C++ | Exception Handling | Question 3
7. C++ | Exception Handling | Question 4
8. C++ | Exception Handling | Question 5
9. C++ | Exception Handling | Question 6