Differences Between Normal and Static Functions in C++

S.A. Abdulla

21/01/2024

Introduction

In C++, you can't directly catch a variable like `catch(k)`. When catching exceptions, you need to specify the type of the exception being caught; you can't use only the variable name to catch.
For example, if you want to catch an exception of type `int`, you would typically catch it by reference:

```
int k;
try {
    // code that may throw an int
} catch (int &k) {
    // handle the exception
}
```

Listing 1: Example of catching an int exception

In C++, a `throw` statement can only throw one exception object at a time. Each `throw` statement typically throws one instance of a particular exception type.

Custom Exception Handling

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

class CustomException {
public:
    int errorCode;
    string errorMessage;

    CustomException(int code, const string& message) : errorCode(code),
        errorMessage(message) {}
};

void foo() {
    throw CustomException(404, "Not Found");
}

int main() {
    try {
        foo();
    }
    catch(const CustomException& e) {
        cout << "Error Code: " << e.errorCode << endl;
        cout << "Error Message: " << e.errorMessage << endl;
    }
    return 0;
}
```

Listing 2: Example of custom exception handling

Handling Multiple Exceptions

If you want to handle multiple exceptions, it has to be done separately. If we use `throw` one after another without catching, only one of the `catch` blocks will execute.

```cpp
#include <iostream>

using namespace std;

// Define the Exception class
class MyException {
public:
    int value;
    bool isPtrException;

    MyException(int val, bool isPtr) : value(val), isPtrException(isPtr) {}
};

// Define the fun function
void fun(int* ptr, int val) {
    if (ptr == nullptr) {
        throw MyException(val, true);
    } else {
        throw MyException(val, false);
    }
}

int main() {
    int y = 0;
    try {
        fun(nullptr, y); // Throws MyException with isPtrException=true
    } catch (MyException &e) {
        if (e.isPtrException) {
            cout << "Caught exception from ptr value: " << e.value << endl;
        } else {
            cout << "Caught exception from val value: " << e.value << endl;
        }
    }

    try {
        int x = 0;
        fun(&x, y); // Throws MyException with isPtrException=false
    } catch (MyException &k) {
        if (k.isPtrException) {
            cout << "Caught exception from ptr value: " << k.value << endl;
        } else {
            cout << "Caught exception from val value: " << k.value << endl;
        }
    }

    return 0;
}
```

Listing 3: Example of handling multiple exceptions