

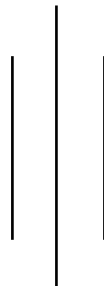


TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING



HIMALAYA COLLEGE OF ENGINEERING CHYASAL, LALITPUR



Lab Report No: - Exception Handling
Title: -08

Submitted by: -
Name: -Aditya Man Shrestha
Roll NO: -HCE081BEI005

Submitted To: -
Department Of
Checked by: -

Date of submission: -

Objectives:

- To understand the concept of Exception handling in C++.

Tools and Libraries Used:

- Programming Language: C++
- IDE: Code::Blocks
- Libraries: include <iostream>, include <string>

Theory:

Exception handling in C++ is a mechanism that allows a program to detect and manage runtime errors gracefully, without crashing. It separates error-handling code from the main logic, improving readability and robustness.

C++ uses three main keywords:

- try: Defines a block of code to test errors.
- throw: Signals the occurrence of an exception.
- catch: Defines a block of code that handles the error.

Syntax:

```
try {  
    throw "Error occurred";  
}  
catch (const char* msg) {  
    cout << msg;  
}
```

You can throw and catch different types: integers, strings, objects, or even custom exception classes.

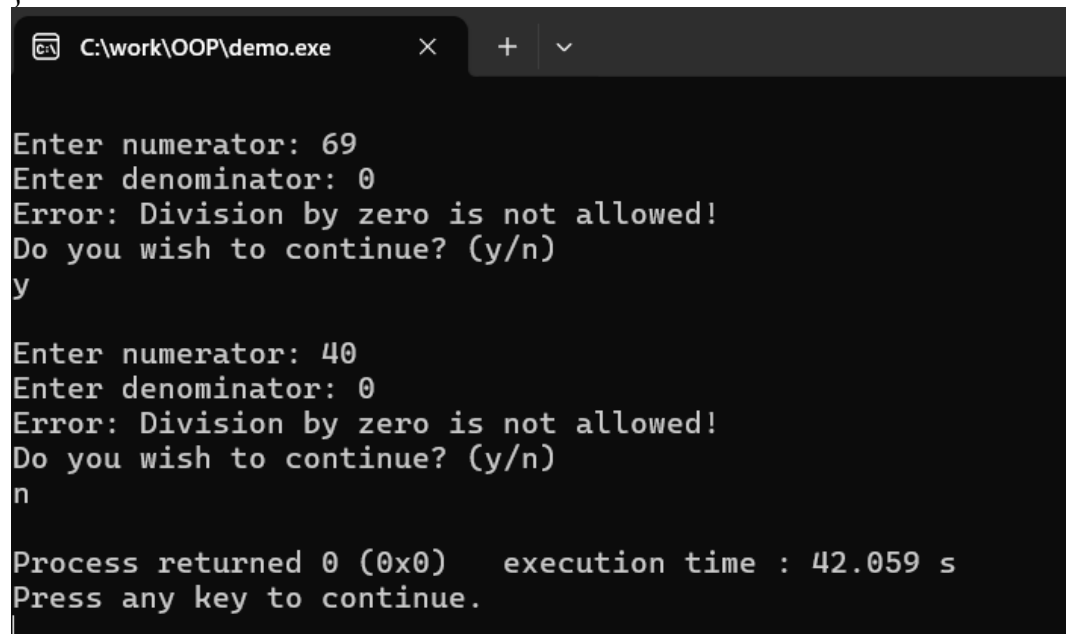
Benefits:

- Handles errors without terminating the program.
- Promotes clean, maintainable code

Lab Assignment

Qn1.

```
#include <iostream>
using namespace std;
int main() {
    double numerator, denominator, result;
    cout << "\nEnter numerator: ";
    cin >> numerator;
    cout << "Enter denominator: ";
    cin >> denominator;
    try {
        if (denominator == 0) {
            throw "Division by zero is not allowed!";
        }
        result = numerator / denominator;
        cout << "Result: " << result << endl;
    }
    catch (const char* msg) {
        cout << "Error: " << msg << endl;
        cout << "Do you wish to continue? (y/n)" << endl;
        char ch;
        cin >> ch;
        if (ch == 'y')
            main();
    }
    return 0;
}
```



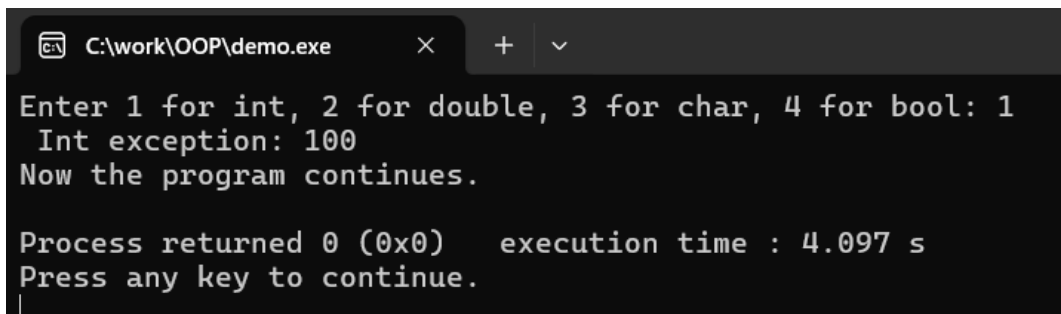
```
C:\work\OOP\demo.exe
Enter numerator: 69
Enter denominator: 0
Error: Division by zero is not allowed!
Do you wish to continue? (y/n)
y

Enter numerator: 40
Enter denominator: 0
Error: Division by zero is not allowed!
Do you wish to continue? (y/n)
n

Process returned 0 (0x0)   execution time : 42.059 s
Press any key to continue.
```

Qn2.

```
#include <iostream>
using namespace std;
int main() {
    try {
        int choice;
        cout << "Enter 1 for int, 2 for double, 3 for char, 4 for bool: ";
        cin >> choice;
        if (choice == 1)
            throw 100;
        else if (choice == 2)
            throw 3.14;
        else if (choice == 3)
            throw 'Z';
        else if (choice == 4)
            throw true;
        else
            throw "Unknown type";
    }
    catch (int e) {
        cout << " Int exception: " << e << endl;
    }
    catch (...) {
        cout << "Here is an exception of unknown or unexpected type!" << endl;
    }
    cout << "Now the program continues." << endl;
    return 0;
}
```



```
C:\work\OOP\demo.exe
Enter 1 for int, 2 for double, 3 for char, 4 for bool: 1
Int exception: 100
Now the program continues.

Process returned 0 (0x0)   execution time : 4.097 s
Press any key to continue.
```

Conclusion:

Understanding exception handling in C++ is essential for writing robust and error resilient programs. Through this lab, we learned how to:

- Use try, throw, and catch blocks to handle runtime errors gracefully.
- Catch and manage different types of exceptions, including custom types.
- Maintain program flow without abrupt termination during unexpected errors.

These concepts are crucial for developing reliable and maintainable applications.

Mastery of exception handling ensures that programs can deal with errors effectively, improving both user experience and code quality.