Himalaya College of Engineering

Advanced C++ Programming Lab Report

Lab 9: Exception Handling in C++

Prepared By : Nawnit Paudel (HCE081BEI024)

Subject : Object-Oriented Programming (OOP)

Program : Bachelor of Electronics, Communication and Information Engineering

Institution : Himalaya College of Engineering

OBJECTIVE

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

BACKGROUND THEORY

Definition

When an issue occurs (like division by zero), an "exception" is **thrown**. This exception is then **caught** by dedicated code, allowing your program to recover or exit gracefully.

It helps keep your main code clean and easily propagate error information.

Syntax

Exception handling in C++ uses three keywords: **try**:

Encloses code that might cause a problem.

```
try {
   // Code that could throw an exception
}
```

throw: Signals an exceptional condition. You can throw numbers, text, or custom error objects.

```
if (value < 0) {     throw "Negative
value not allowed!";
}
```

catch: Handles exceptions thrown from a try block. Each catch block handles a specific type of exception.

```
catch (const char* errorMessage) {
   // Handle text message error
} catch (int
errorCode) {
   // Handle integer error code
} catch
(...) {
   // Catch any other exception
}
```

3. Types of Exceptions

You can throw:

Custom Exceptions: Error types you create using C++ classes for specific details. class MyError : public std::exception { public:

std::string message;

MyError(const std::string& msg) : message(msg) {}

};

Standard Exceptions: Built-in C++ exceptions for common problems (e.g., std::runtime_error, std::logic_error, std::out_of_range, std::bad_alloc).

4. Simple Example

```
#include <iostream>
#include <string>

double divide(double numerator, double denominator) {
if (denominator == 0) {         throw "Error: Cannot
divide by zero!";
   }
   return numerator / denominator;
}

int main() {
   try {         double result1 = divide(10.0, 2.0);
std::cout << "10 / 2 = " << result1 << std::endl;

      double result2 = divide(5.0, 0.0);         std::cout << "This
line will not be executed." << std::endl;
   }
   catch (const char* errorMessage) {
      std::cerr << "Caught an error: " << errorMessage << std::endl;
```

```
    }

    std::cout << "Program continues after error handling." << std::endl;

    return 0;
}
```

Explanation of the Example:

1. divide throws an error if the denominator is zero.
2. main calls divide within a try block.
3. When divide(5.0, 0.0) throws, the try block stops. 4. The catch block for const char* handles the error.
5. Program execution resumes after the catch block.

LAB ASSIGNMENTS

1. Write a C++ program to handle divide-by-zero exception using try-catch block. Input two numbers. If the denominator is zero, throw and catch an exception.

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

int main() {
    float numerator, denominator;

    cout << "Enter numerator: ";
    cin >> numerator;

    cout << "Enter denominator: ";
    cin >> denominator;

    try {
```

```
        if (denominator == 0) {
            throw "Division by zero exception!";
        }
        float result = numerator / denominator;
        cout << "Result: " << result << endl;
    } catch (const char* msg) {
        cout << "Error: " << msg << endl;
    }

    return 0;
}
```

2. Write a C++ program to demonstrate multiple catch blocks handling different data types. Throw and handle int, char, and string type exceptions in separate catch blocks.

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

int main() {
    int choice;
    cout << "Choose exception to throw:" << endl;
    cout << "Enter your choice:\n1. int\n2. char\n3. string" << endl;
    cin >> choice;

    try {
        if (choice == 1) {
            throw 100;
        } else if (choice == 2) {
            throw 'A';
        } else if (choice == 3) {
```

```
            throw string("This is a string exception!");
        } else {
            cout << "Invalid choice. No exception thrown." << endl;
        }
    } catch (int e) {
        cout << "Caught an int exception: " << e << endl;
    } catch (char e) {
        cout << "Caught a char exception: " << e << endl;
    } catch (string& e) {
        cout << "Caught a string exception: " << e << endl;
    }

    return 0;
}
```

3. Write a program using catch-all handler (catch(...)) to handle any kind of exception. Illustrate a case where an unexpected data type is thrown and caught generically.

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

int main() {
    int choice;
    cout << "Choose exception to throw:" << endl;
    cout << "1. int" << endl;
    cout << "2. string" << endl;
    cout << "3. Other" << endl;
    cout << "Enter your choice: ";
    cin >> choice;

    try {
        if (choice == 1) {
            throw 42;
        } else if (choice == 2) {
            throw string("A string exception!");
        } else if (choice == 3) {
            throw 3.14159;
        } else {
```

```
            cout << "Invalid choice. No exception thrown." << endl;
        }
    } catch (int e) {
        cout << "Caught int exception: " << e << endl;
    } catch (const string& e) {
        cout << "Caught string exception: " << e << endl;
    } catch (...) {
        cout << "Caught an exception of unknown or unexpected type!" << endl;
    }

    return 0;
}
```

DISCUSSION:

Exception handling in C++ helps programs deal with errors that happen while running. Instead of crashing, the program can catch these errors using try, throw, and catch. This way, the program can handle different kinds of problems and keep working or stop safely.

CONCLUSION:

Exception handling makes C++ programs safer and more reliable by allowing errors to be caught and managed properly. Instead of the program crashing unexpectedly, it can respond to problems in a controlled way, such as showing a message or trying an alternative action. This helps improve user experience and makes the program easier to maintain and debug. Overall, exception handling is an important tool for writing strong and stable C++ programs.