Objective

1. To understand the concept of exception handling.
2. To learn how to use try, catch, finally or except blocks to handle run-time errors.
3. To write programs that gracefully handle different types of exceptions such as division by zero, invalid input, or file not found.

Theory

Exception Handling is a mechanism that allows a program to deal with runtime errors or abnormal conditions (called exceptions) in a controlled manner, preventing the program from crashing unexpectedly. It uses specific constructs like try, catch (or except), and finally to handle and respond to errors gracefully.

C++ Exception Handling Syntax

```
try {
   // Code that might throw an exception
}
catch (ExceptionType e) {
   // Code to handle the exception
}
```

Example:

```
try {
   int result = 10 / 0;
}
catch (exception &e) {
   cout << "Exception caught: " << e.what();
}
```

Common C++ Exceptions with Syntax

1. Division by Zero

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

int main() {
   int a = 10, b = 0;
   try {
      if (b == 0)
         throw "Division by zero not allowed";
      int c = a / b;
```

```
    }
    catch (const char* msg) {
        cout << "Exception: " << msg << endl;
    }
    return 0;
}
```

2. Array Index Out of Bound

```
#include <iostream>
using namespace std;
int main() {
    int arr[3] = {1, 2, 3};
    try {
        if (4 >= 3)
            throw "Array index out of bounds";
        cout << arr[4];
    }
    catch (const char* msg) {
        cout << "Exception: " << msg << endl;
    }
    return 0;
}
```

3. Null Pointer Dereferencing

```
#include <iostream>
using namespace std;
int main() {
    int* ptr = nullptr;
    try {
        if (ptr == nullptr)
            throw "Null pointer dereferenced";
        cout << *ptr;
    }
    catch (const char* msg) {
        cout << "Exception: " << msg << endl;
    }
    return 0;
}
```

4. File Not Found Exception (Manual Throw)

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

int main() {
    ifstream file("nonexistent.txt");
    try {
```

```
        if (!file)
            throw "File not found!";
        cout << "File opened successfully." << endl;
    }
    catch (const char* msg) {
        cout << "Exception: " << msg << endl;
    }
    return 0;
}
```

5. Custom Exception Class

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

class MyException : public exception {
public:
    const char* what() const throw() {
        return "Custom Exception occurred";
    }
};
int main() {
    try {
        throw MyException();
    }
    catch (MyException& e) {
        cout << "Exception: " << e.what() << endl;
    }
    return 0;
}
```

Multiple Catch Blocks

```
try {
    throw 10; // throw different types for testing
}
catch (int e) {
    cout << "Integer exception: " << e << endl;
}
catch (const char* msg) {
    cout << "String exception: " << msg << endl;
}
```

Catch-All Handler

```
try {
    throw 3.14;
}
catch (...) {
```

```
    cout << "Unknown exception caught." << endl;
```