

# Exception Handling

Ali Haider

[syedalihaider.ciit@gmail.com](mailto:syedalihaider.ciit@gmail.com)

Department of Computer Science IUB

# Overview

- Exception
- Use of try, catch and throw
- Exception Handling

# Exception

- An exception is a problem that arises during the execution of a program.
- A C++ exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero.
- Exceptions provide a way to transfer control from one part of a program to another.
- C++ exception handling is built upon three keywords: **try**, **catch**, and **throw**.

# Use of try, catch and throw

- **throw** – A program throws an exception when a problem shows up. This is done using a **throw** keyword.
- **catch** – A program catches an exception with an exception handler at the place in a program where you want to handle the problem.
- The **catch** keyword indicates the catching of an exception.
- **try** – A **try** block identifies a block of code for which particular exceptions will be activated.
- It's followed by one or more catch blocks.

# Exception Handling

- Assuming a block will raise an exception, a method catches an exception using a combination of the **try** and **catch** keywords.
- A try/catch block is placed around the code that might generate an exception.
- Code within a try/catch block is referred to as protected code, and the syntax for using try/catch as follows –

```
try {  
    // protected code  
} catch( ExceptionName e1 ) {  
    // catch block  
} catch( ExceptionName e2 ) {  
    // catch block  
} catch( ExceptionName eN ) {  
    // catch block  
}
```

# Throwing Exceptions

- Exceptions can be thrown anywhere within a code block using **throw** statement.
- The operand of the throw statement determines a type for the exception and can be any expression and the type of the result of the expression determines the type of exception thrown.

```
double division(int a, int b) {  
    if( b == 0 ) {  
        throw "Division by zero condition!";  
    }  
    return (a/b);  
}
```

# Catching Exceptions

- The **catch** block following the **try** block catches any exception.
- You can specify what type of exception you want to catch and this is determined by the exception declaration that appears in parentheses following the keyword catch.

```
try {  
    // protected code  
} catch( ExceptionName e ) {  
    // code to handle ExceptionName exception  
}
```

# Example

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

double division(int a, int b) {
    if( b == 0 ) {
        throw "Division by zero condition!";
    }
    return (a/b);
}

int main () {
    int x = 50;
    int y = 0;
    double z = 0;

    try {
        z = division(x, y);
        cout << z << endl;
    } catch (const char* msg) {
        cerr << msg << endl;
    }

    return 0;
}
```



# Example

- try {
- int age = 15;
- if (age > 18) {
- cout << "Access granted - you are old enough.";
- } else {
- throw (age);
- }
- }
- catch (int myNum) {
- cout << "Access denied - You must be at least 18 years old.\n";
- cout << "Age is: " << myNum;
- }

## Cont...

- We use the try block to test some code: If the age variable is less than 18, we will throw an exception, and handle it in our catch block.
- In the catch block, we catch the error and do something about it.
- The catch statement takes a parameter:
- in our example we use an int variable (myNum) (because we are throwing an exception of int type in the try block (age)), to output the value of age.

# Handle Any Type of Exceptions

- If you do not know the throw type used in the try block, you can use the "three dots" syntax (...) inside the catch block, which will handle any type of exception:
- ```
try {  
    int age = 15;  
    if (age > 18) {  
        cout << "Access granted - you are old enough.";  
    } else {  
        throw 505;  
    }  
}  
catch (...) {  
    cout << "Access denied - You must be at least 18 years old.\n";  
}
```