Exceptions

Course Code: CSC1102 &1103 Course Title: Introduction to Programming

Dept. of Computer Science
Faculty of Science and Technology

Lecturer No:	**13**	**Week No:**	**10(1X1.5)**	**Semester:**	
Lecturer:	*Name & email*				

After completing this module, the students will be able to:

- Understand situations when exceptions are thrown
- Understand the concept of exceptions handling
- Identify situations when exceptions are thrown
- Add code to handle and throw exceptions
- Write his/her own exception classes

After completing this module, the students will be able to:

- Pass additional information in exceptions
- Handle file-related exceptions
- Throw exceptions in constructors
- Use exceptions in real programs
- Simplify exception throwing

How to get into trouble?

There is no code free of errors.

One of Murphy's laws even states that

"An error-free code is actually only one that has been sufficient tested"

Take a look at the example →

```
#include <iostream>
using namespace std;
int main(       ) {
    float a, b;
    cin >> a;
    cin >> b;
    cout << a / b << endl;
    return 0;
}
```

Compile and run the program and enter "0" (zero) as the second value. What happens then?

Take a look at the example →

```
#include <iostream>
using namespace std;
int main(void) {
    float a, b;
    while(cin >> a) {
    cin >> b;
    cout << a / b << endl;
}
   return 0; }
```

How to get out of trouble ?

- When something goes wrong in a program, an exception arises.
- Some exceptions arise automatically, completely out of our control, while others may be created at our request.

How to get out of trouble ? (CONTD.)

- The part of the code that may cause problems needs to be marked
- When an exception arises, the execution of the block is terminated

What is exceptions?

- 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.

What is exceptions?

Exceptions

- Indicate problems that occur during a program's execution
- Occur infrequently

Exception handling

- Can resolve exceptions
  - Allow a program to continue executing or
  - Notify the user of the problem and
  - Terminate the program in a controlled manner
- Makes programs robust and fault-tolerant

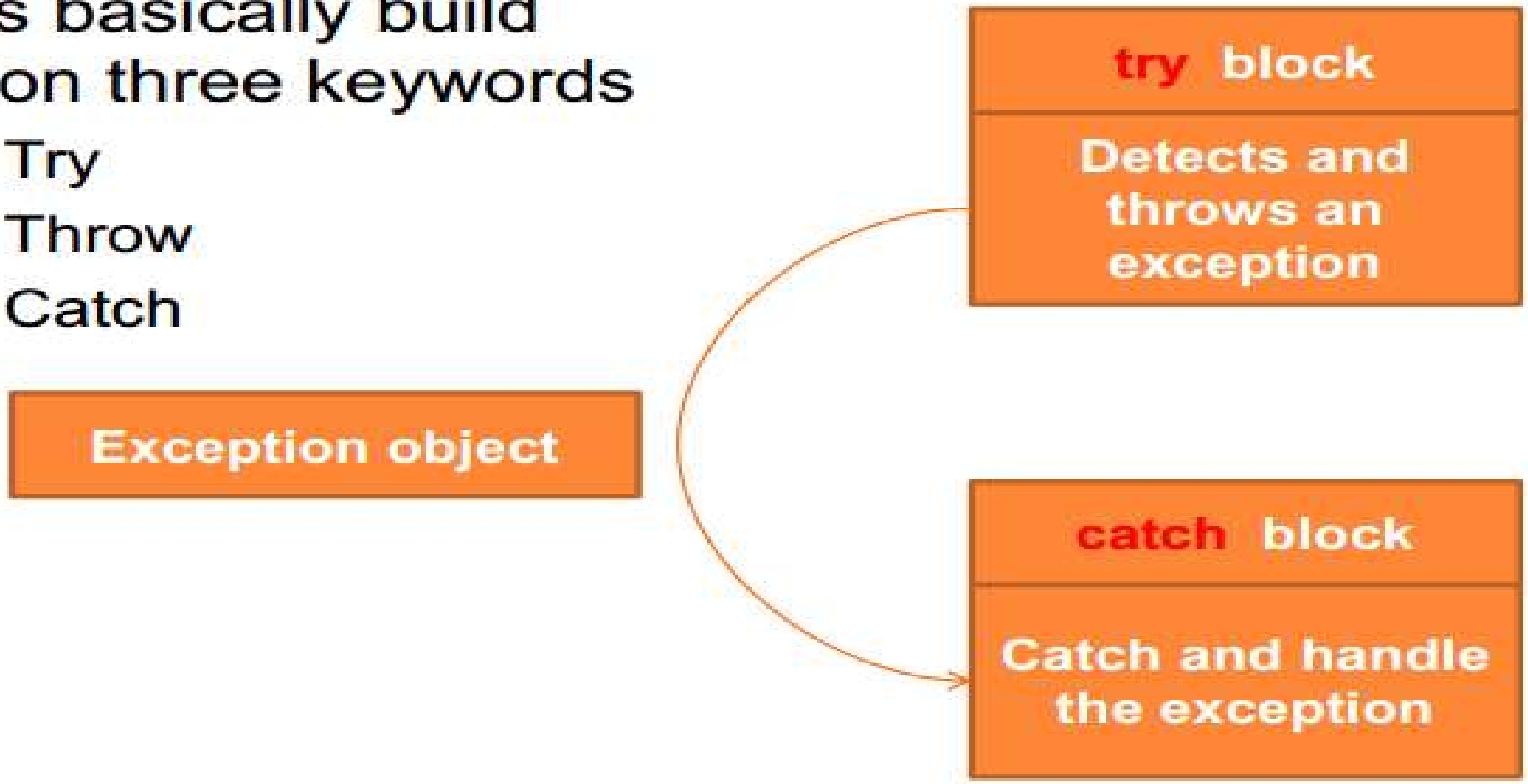
Why Exception Handling?

- Following are main advantages of exception handling over traditional error handling.

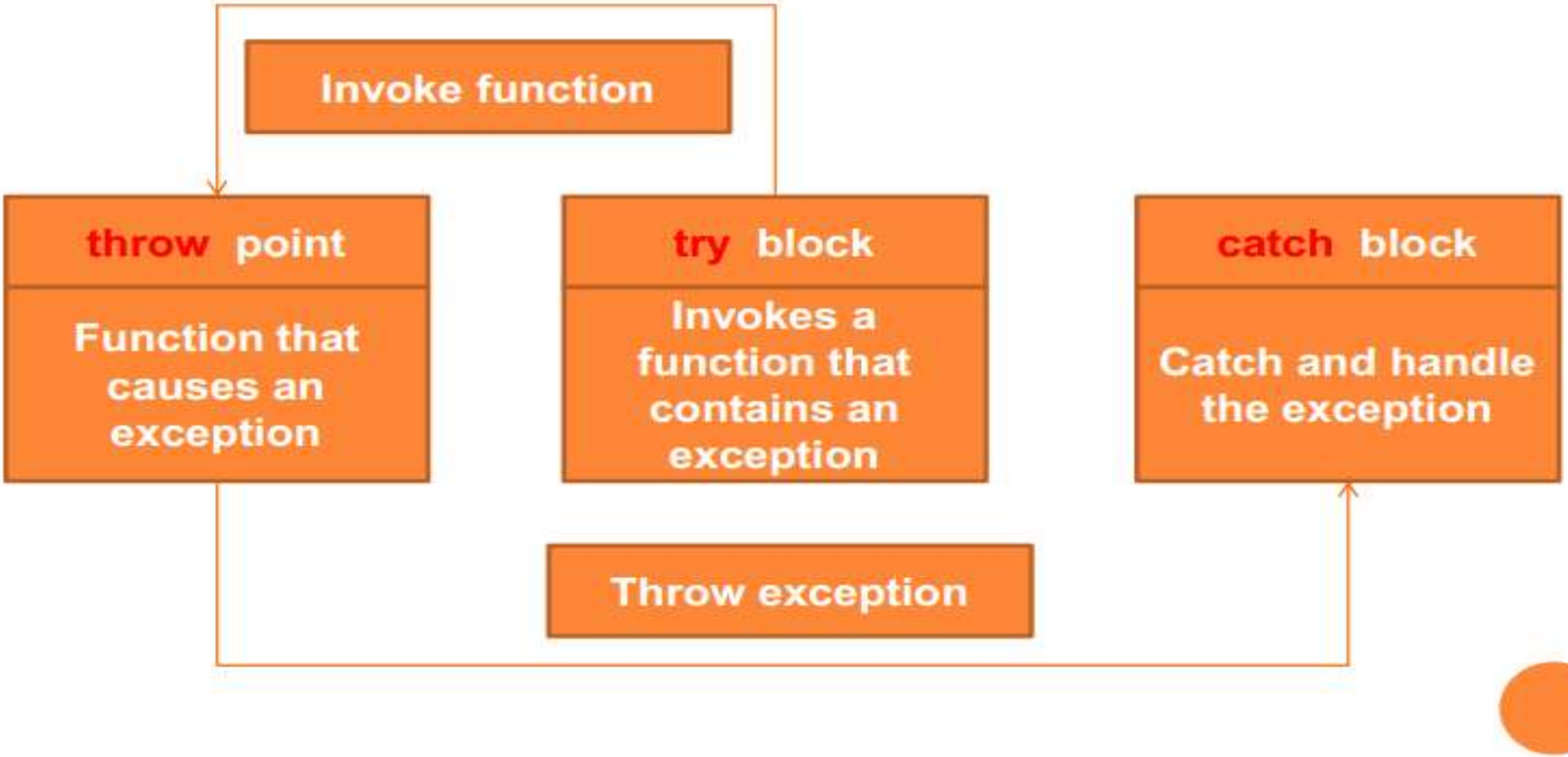
1) *Separation of Error Handling code from Normal Code*
2) *Functions/Methods can handle any exceptions they choose*
3) *Grouping of Error Types*

EXCEPTION HANDLING MECHANISM

- It is basically build upon three keywords
  - Try
  - Throw
  - Catch



EXCEPTION HANDLING MECHANISM (CONT…)



What is exceptions?

- 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.
- **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.

Throw, catch, try

- **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.

Throw, catch, try

- 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.
- If you write it this way:

Throwing Exceptions

- Following is an example of throwing an exception when dividing by zero condition occurs −

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

Anatomy of an exception object

- The exception class is very modest. In fact it defines only three components:
  - a constructor
  - a virtual destructor, originally empty
  - a virtual function called what which returns the C-style string

```
virtual char* what()
```

  - The what function provides a text (more or less verbose) describing the nature and cause of the exception.

Thank You