**CHAPTER 6**

EXCEPTION HANDLING

\*\* Exceptions are runtime anomalies (irregularities) or an unusual condition that a program may face.

\*\* Exceptions are neither syntax error nor logical error (wrong solution).

\*\* 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 θ 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. Visually written inside the try block

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.

♣ Handling or catching the exception thrown by the throw statement.

♣ When throw is true, catch is called.

try:

♣ Block that may rise exception. A try block identifies a block of code for which particular exceptions will be activated. It's followed by one or more catch blocks.

***[ Note by -Jannatul Ferdous Umama(Bristy)]***

**Throwing Exception**

\*\* Exceptions can be thrown anywhere within a code block using throw statements.

\*\* The operand of the throw statements 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 Exception**

\*\* Assuming a block will raise an exception, a method catches an exception using a combination of the try and catch keywords.

\*\* You can list down multiple catch statements to catch different type of exceptions in case your try block raises more than one exception in different situations.

Try

{

// protected code

}

catch( ExceptionName e1 )

{

// catch block

}

catch( ExceptionName e2 )

{

// catch block

}

catch( ExceptionName eN )

{

// catch block

}

**Catching Exception**

\*\* A try block must be followed by at least one catch block (otherwise compiler error occurs).

\*\* If there are more than one catch block (multiple catch block) then the catch which meets the data type is executed.

\*\* If there is no catch for a corresponding throw then following system functions are called Terminate() -> abort()

\*\* If you want to specify that a catch block should handle any type of exception that is thrown in a try block, you must put an ellipsis, ..., between the parentheses enclosing the exception declaration as follows

try

{

// protected code

}

catch(...)

{

// code to handle any exception

}

***[ Note by -Jannatul Ferdous Umama(Bristy)]***