Handling exceptions (Try / Catch / Finally) in C

An Exception is an issue that occurs during the execution of a program. An Exception in C # is a response to an exception situation that occurs while a program is running, such as dividing by zero.

Exception is an issue that appears during program execution. An Exception in C # is a response to an exception situation that occurs while the program is running, for example by dividing by zero.

Exception provides a way to pass control from one part of a program to another. Just like the exception handling in PHP, Exception Handling in C # is built on four keywords: **try, catch, finally,** and **throw** .

1. **try** : A try block identifies a block of code where specific exceptions are activated. It is followed by one or more catch blocks.
2. **catch** : A program catches an Exception with an Exception Handler in place in a program where you want to handle that problem. The catch keyword in C # shows how to catch an exception.
3. **finally** : A finally block is used to execute a given set of commands, whether or not an exception is thrown or not thrown. For example, if you open a file, it must be closed, otherwise an exception will be created.
4. **throw** : A program throws an exception when a problem occurs. This is done using the **throw** keyword in C #.

Syntax

Suppose a block creates an Exeption, a method of catching an exception by using a combination of try and catch keywords. A try / catch block is placed around the code that can create an exception. The code inside a try / catch block is treated as protected code, and the syntax for using try / catch in C # is as follows:

```
try {
// commands may cause an exception (exception)
} catch (foreign_folder_name e1) {
// code to handle errors
} catch (foreign_folder_name e2) {
// code to handle errors
} catch (foreign_folder_name) {
// code to handle errors
} finally {
// commands to be executed
}
```

You can list many catch commands to catch different exception types in case your try block appears more than one exception in different situations.

Exception class in C

Exceptions in C # are represented by classes. The Exception classes in C # are primarily inherited directly or not directly from the **System.Exception** class in C #. Some Exception classes inherit from the System.Exception class are **System.ApplicationException** and **System.SystemException** classes.

The **System.ApplicationException** class supports the exceptions created by application programs. Therefore, the exceptions defined by the programmer should inherit from this class.

The **System.SystemException** class is the base class for all pre-defined system exception.

The following table provides some classes of pre-defined Exception inherited from the *Sytem.SystemException* class *in C #* :

Exception Class DescriptionSystem.IO.IOException Handling error I / O. System.IndexOutOfRangeException Handling errors created when a method references an index outside the array array. System.ArrayTypeMismatchException Handling errors created when type is not suitable for array type. System.NullReferenceException Handling errors created from referencing a null object. System.DivideByZeroException Handling errors created when dividing by 0. System.InvalidCastException Error handling generated during casting. System.OutOfMemoryException Error handling is created from the lack of free memory. System.StackOverflowException Error handling generated from stack overflow.

Exception handling (Exception Handling) in C

C # provides a highly structured solution for exception handling in the form of try and catch blocks. Using these blocks, the main commands of the program are separated from the error handling commands in C #.

These exception handling blocks are implemented by using **try, catch** and **finally** keywords in C #. For example, throw an exception when dividing by 0.

```
 using System ; namespace VdXuLyNgoaiLe { class PhepChia { int  result ; PhepChia
?t qu?
: {0}" ,  result ); } } static void Main ( string []  args ) { PhepChia  d  = new
```

When compiling and running the above C # program will produce the following results:

```
  Exception caught: System.DivideByZeroException: Attempted to divide by zero.
  at VdXuLyNgoaiLe.PhepShare.chia (System.Int32 so1, System.Int32 so2) [0x00000]
  Result: 0
```

Create User-Defined Exception in C

You can also define exceptions. The User-Defined Exception classes are inherited from the **ApplicationException** class in C #. The following example illustrates this:

Creating 3 classes named in turn is as follows:

Temperature class

```
 using System ; namespace VdDinhNghiaException { class KtraNhietDo { static void
?c nhi?t ?? b?ng 0!" )); } else { Console . WriteLine ( "Nhi?t ??
```

```
: {0}" ,  temperature ); } } }
```

Compiling and running the above C # program will produce the following results:

```
  TempIsZeroException: Temperature level is 0!
```

Throwing Object in C

You can throw an object if it: either directly or indirectly inherits from the System.Exception class in C #. You can use a throw command in the catch block to throw that presence object:

```
  Catch (Exception e)
  {
  .
  Throw e
  }
```

According to Tutorialspoint

Last lesson: Regular Expression in C #

Next article: File I / O in C #