

EJERCICIOS MÓDULO 4 Excepciones

Introducción

Para tratar este tema vamos a reaprovechar algunos de los ejercicios de los bloques anteriores, agregando a algunos de ellos las sentencias try...catch. Usaremos algunas excepciones predefinidas y otras las haremos personalizadas.

Antes de empezar con los ejercicios, vamos a recordar el orden de ejecución de un programa que gestiona excepciones.

-Si el usuario hace todo correctamente y no hay ningún problema interno del sistema, el orden de ejecución será:

```
try  
finally
```

-Si el usuario comete algún fallo o bien sucede algún error interno del sistema, el orden de ejecución del programa será:

```
try  
catch (Exception e)  
finally
```

Excepciones predefinidas

Todos los lenguajes de programación actuales vienen con un conjunto de excepciones predefinidas que saltan al cometerse determinadas irregularidades provocadas tanto por el usuario como a nivel interno del sistema. A continuación tienes algunas de estas excepciones para nuestro caso particular (C#):

DivideByZeroException: el divisor (número por el que dividimos) es cero.

FormatException: el formato de entrada del usuario no coincide con el esperado. Por ejemplo, estamos esperando un valor numérico y el usuario introduce una cadena de caracteres.

ArgumentNullException: se produce al intentar hacer referencia a un objeto cuyo valor es null.

OutOfMemoryException: se produce cuando el intento de asignar memoria mediante el operador new da un error. Esto indica que la memoria disponible se ha agotado.

OverflowException: se produce cuando una operación aritmética produce un desbordamiento, esto es, que el resultado sea un número que ocupe más memoria de la que se le había reservado a priori (por ejemplo, un número tan grande que no quepa en un int, sino que se debería usar un long).

Problema 1

Haz un programa que lea un texto por teclado, y lo pase a número (usa para ello el método `int.Parse()`). Prueba con los siguientes datos:

- 15
- Dieciséis

¿Qué sucede en cada caso, y por qué?

Problema 2

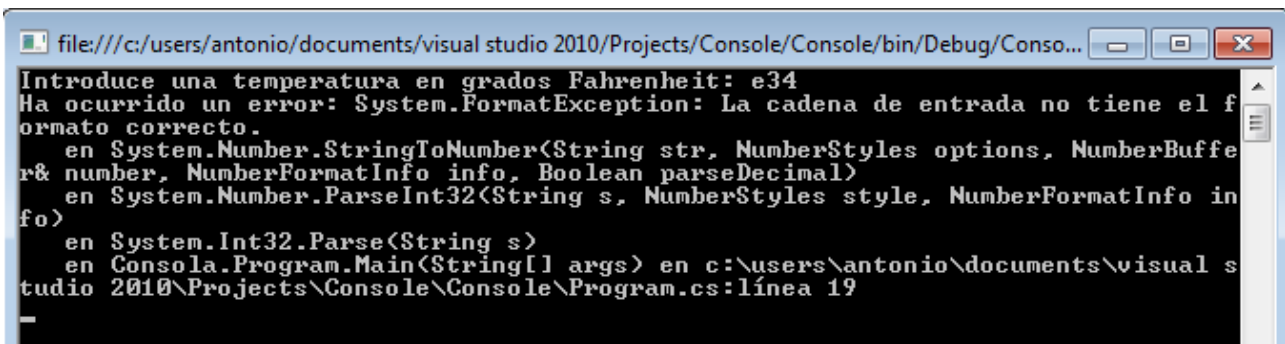
Controla la excepción del ejercicio anterior, de manera que el programa de un aviso al usuario de “número incorrecto” si sucede un error.

Ejercicio 3

Haz un programa en modo Consola que realice la traducción de grados Fahrenheit a Grados Celsius. La formula de conversión es la siguiente:

$$\text{gradosCelsius} = (\text{gradosFahrenheit} - 32) * 5 / 9;$$

En caso de introducir un valor numérico erróneo, el programa debe de ser capaz de capturar el error y mostrar el siguiente mensaje:



```
file:///c:/users/antonio/documents/visual studio 2010/Projects/Console/Console/bin/Debug/Conso...
Introduce una temperatura en grados Fahrenheit: e34
Ha ocurrido un error: System.FormatException: La cadena de entrada no tiene el formato correcto.
   en System.Number.StringToNumber(String str, NumberStyles options, NumberBuffer& number, NumberFormatInfo info, Boolean parseDecimal)
   en System.Number.ParseInt32(String s, NumberStyles style, NumberFormatInfo info)
   en System.Int32.Parse(String s)
   en Console.Program.Main(String[] args) en c:/users/antonio/documents/visual studio 2010/Projects/Console/Console/Program.cs: línea 19
```

Problema 4

Haz un programa que divida dos números, y controle que el divisor sea distinto de 0 mediante la captura de una determinada excepción.

[La división entre un denominador 0 provoca la excepción `DivideByZeroException`.](#)

Ejercicio 5

El siguiente programa pide dos números al usuario y realiza la división del primer número entre el segundo sin control de excepciones:

```
static void Main(string[] args)
{
    int num1, num2;
    double divi;
    string temp1, temp2;
    Console.Write("Introduce primer numero: ");
    temp1 = Console.ReadLine();
    if (temp1 == "") temp1 = null;
    Console.Write("Introduce segundo numero: ");
    temp2 = Console.ReadLine();
    if (temp2 == "") temp2 = null;
    num1 = int.Parse(temp1);
    num2 = int.Parse(temp2);
    divi = (double)(num1*1.0 / num2);
    Console.WriteLine(num1 + "/" + num2 + "=" + divi);
    Console.ReadKey();
}
```

a) A través de diferentes pruebas examina las excepciones que se pueden producir al introducir los siguientes valores:

Num1	Num2	Excepciones
edu	5	FormatException
5	0	DivideByZeroException
"" (enter)	5	ArgumentNullException
100000000000	1	OverflowException

b) Captura y trata las excepciones producidas anteriormente con el objetivo de que el programa no finalice de forma abrupta su ejecución y pueda mostrar un mensaje de error al usuario.

c) Examina el papel de la clausula finally. Introduce aquí la instrucción Console.ReadKey(). Observa si el programa realiza la parada tanto si se realiza correctamente la división como si se produce una excepción.

Ejercicio 6

Dada la estructura del ejercicio anterior, consigue que el programa siga preguntando hasta que el usuario introduzca correctamente los números a operar (que sean realmente números), a partir de la estructura try catch. Se debe poder realizar correctamente la secuencia de división sin ningún tipo de problema, repitiéndola tantas veces como sea necesario:

Ejercicio 7

Haz una función que calcule el factorial de un número. Por ejemplo, el factorial de 5 es: 5 x 4 x 3 x 2 x 1. El de 3 es 3 x 2 x 1. Si el usuario intenta un factorial de un número < que 0, o que >15, la función debe lanzar una excepción del tipo `ArgumentOutOfRangeException`, indicando el rango correcto para realizar la operación.

Ejercicio 8

Modifica el ejercicio anterior de manera que ahora la función factorial lance una excepción inventada por nosotros, `TePasasteException` derivada de la clase `Exception`.

Ejercicio 9

Dada la siguiente aplicación en WPF, se pide modificarla de manera que realice la misma operativa pero sin utilizar excepciones. Se deben realizar las pertinentes comprobaciones con el objetivo de evitar las siguientes excepciones:

FormatException → Las cadenas introducidas deben de ser números

InvalidOperationException → Se debe seleccionar una operación

DivideByZeroException → El denominador no debe de ser 0 en caso de división

OverflowException → La longitud de los números no debe superar 7 dígitos.

The screenshot shows a WPF application window titled "Excepciones". Inside the window, there are two input fields labeled "Operando 1" and "Operando 2". Between these fields is a container with five radio buttons for selecting an operation: "+ Suma", "- Resta", "* Producto", "/ Division", and "% Resto". Below the operation selection is a "Calculo" button. At the bottom of the window, there are two "Resultado" labels, each followed by an input field for the result. A "Salir" button is located at the bottom left of the window.