

Centro Universitario de Ciencias Exactas e Ingenierías

CUCEI



Computación Tolerante a Fallas

2023B/D06

Profesor: López Franco Michel Emanuel

Otras Herramientas para manejar errores

Gutiérrez Galán Ruben Alejandro

Código: 214798315

Contenido

Introducción:	3
Información general sobre las excepciones:	3
Manejo de excepciones en C#:.....	3
Otras formas de manejar errores en C#:.....	4
Programa ejemplo en ejecución (uso de try-catch y throw):.....	5
Con try catch:	5
Con throw:.....	5
Corriendo con normalidad:	6
Conclusión:	6
Bibliografía:	6

Introducción:

El manejo de errores, también conocido como manejo de excepciones, es una práctica fundamental en la programación que implica prever y tratar situaciones excepcionales o inesperadas que puedan ocurrir durante la ejecución de un programa. Estas situaciones excepcionales, llamadas "excepciones" o "errores", pueden ser causadas por diversos factores, como entradas incorrectas, problemas de red, falta de recursos, problemas de lógica en el código, entre otros.

Información general sobre las excepciones:

Las excepciones tienen las siguientes propiedades:

- Las excepciones son tipos que derivan en última instancia de System.Exception.
- Use un bloque try alrededor de las instrucciones que pueden producir excepciones.
- Una vez que se produce una excepción en el bloque try, el flujo de control salta al primer controlador de excepciones asociado que está presente en cualquier parte de la pila de llamadas. En C#, la palabra clave catch se utiliza para definir un controlador de excepciones.
- Si no hay ningún controlador de excepciones para una excepción determinada, el programa deja de ejecutarse con un mensaje de error.
- No detecte una excepción a menos que pueda controlarla y dejar la aplicación en un estado conocido. Si se detecta System.Exception, reinícielo con la palabra clave throw al final del bloque catch.
- Si un bloque catch define una variable de excepción, puede utilizarla para obtener más información sobre el tipo de excepción que se ha producido.
- Las excepciones puede generarlas explícitamente un programa con la palabra clave throw.
- Los objetos de excepción contienen información detallada sobre el error, como el estado de la pila de llamadas y una descripción de texto del error.
- El código de un bloque finally se ejecuta incluso si se produce una excepción. Use un bloque finally para liberar recursos, por ejemplo, para cerrar las secuencias o los archivos que se abrieron en el bloque try.
- Las excepciones administradas de .NET se implementan en el mecanismo de control de excepciones estructurado de Win32.

Manejo de excepciones en C#:

Las características de control de excepciones del lenguaje C# le ayudan a afrontar cualquier situación inesperada o excepcional que se produce cuando se ejecuta un programa. El control de excepciones usa las palabras clave try, catch y finally para intentar realizar acciones que pueden no completarse correctamente, para controlar errores cuando decide que es razonable hacerlo y para limpiar recursos más adelante. Las excepciones las puede generar Common Language Runtime (CLR), .NET, bibliotecas de terceros o el código de aplicación. Las excepciones se crean mediante el uso de la palabra clave throw.

En muchos casos, una excepción la puede no producir un método al que el código ha llamado directamente, sino otro método más bajo en la pila de llamadas. Cuando se genera una excepción, CLR desenreda la pila, busca un método con un bloque catch para el tipo de excepción específico y

ejecuta el primer bloque catch que encuentra. Si no encuentra ningún bloque catch adecuado en cualquier parte de la pila de llamadas, finalizará el proceso y mostrará un mensaje al usuario.

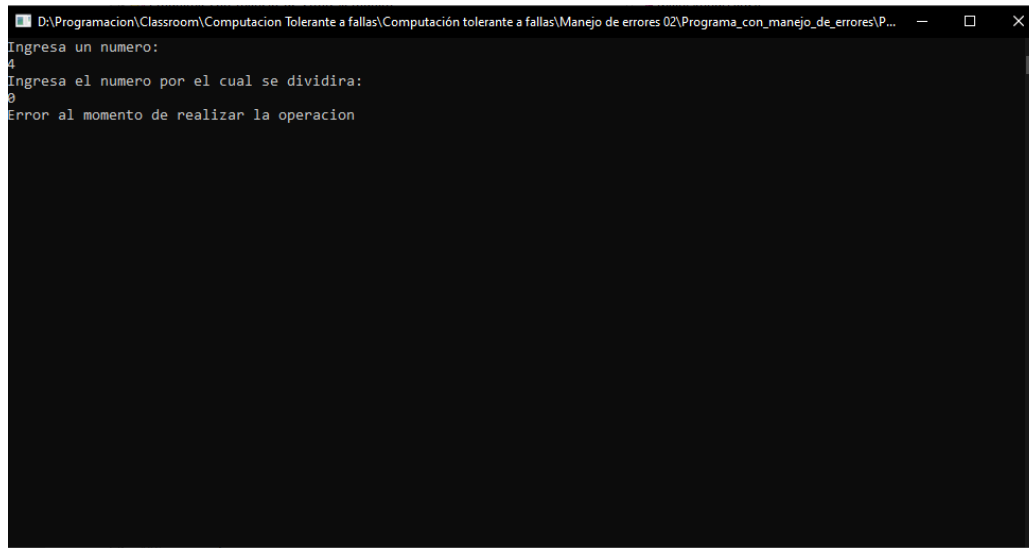
Otras formas de manejar errores en C#:

- **FluentValidation:** Esta biblioteca te permite definir reglas de validación de manera declarativa y fluida. Puedes usarla para validar datos de entrada y evitar errores antes de que ocurran.
- **Aspect-Oriented Programming (AOP) Frameworks:** Frameworks como PostSharp permiten la programación orientada a aspectos, donde puedes encapsular aspectos transversales, como manejo de errores, en aspectos separados, lo que puede mejorar la modularidad y reutilización del código.
- **Serilog / NLog:** Estas son bibliotecas de registro (logging) que ofrecen una forma más avanzada y flexible de registrar información sobre el funcionamiento de tu aplicación. Pueden ayudarte a rastrear y diagnosticar errores de manera más eficiente.
- **Polly:** Polly es una biblioteca de manejo de políticas de manejo de fallas y reintentos. Puede ser útil para manejar de manera más elegante y automatizada las fallas temporales en operaciones que puedan lanzar excepciones.
- **AutoMapper:** Si trabajas con mapeo de objetos entre capas de tu aplicación, AutoMapper te ayuda a hacerlo de manera más eficiente y reducir posibles errores de mapeo manual.
- **Validation Attributes:** Puedes usar atributos de validación personalizados en las propiedades de tus clases para validar datos automáticamente. Esto puede ayudarte a evitar errores de validación.
- **FluentAssertions:** Esta biblioteca de pruebas unitarias permite realizar afirmaciones más legibles y expresivas en tus pruebas. Puede ayudarte a verificar el comportamiento esperado y detectar problemas más rápidamente.
- **Code Contracts:** Aunque no es una extensión tan común, Code Contracts te permite definir contratos en tu código que especifican las condiciones que deben cumplirse en diferentes puntos. Esto puede ayudar a prevenir errores en tiempo de ejecución.
- **Application Insights / New Relic:** Estas son herramientas de monitorización y seguimiento de aplicaciones en tiempo real que pueden ayudarte a identificar problemas y errores en tu aplicación en producción.
- **Enterprise Library Exception Handling Block:** Esta es una parte de Microsoft Enterprise Library que proporciona un enfoque estructurado para el manejo de excepciones y errores en aplicaciones empresariales.

[5]

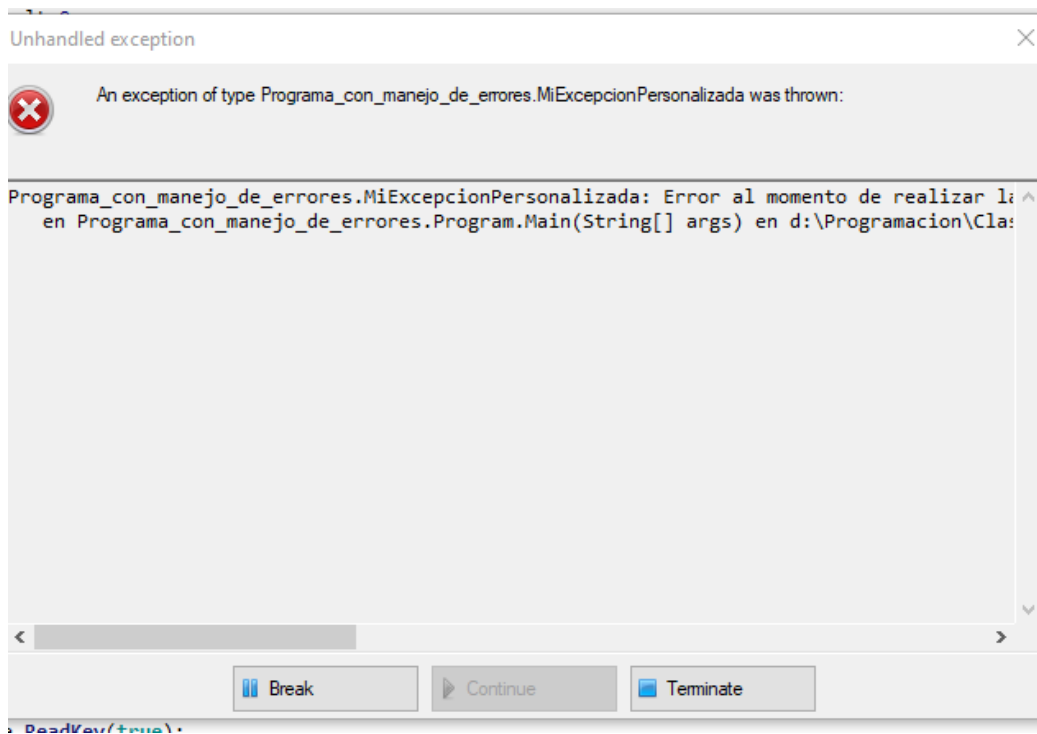
Programa ejemplo en ejecución (uso de try-catch y throw):

Con try catch:

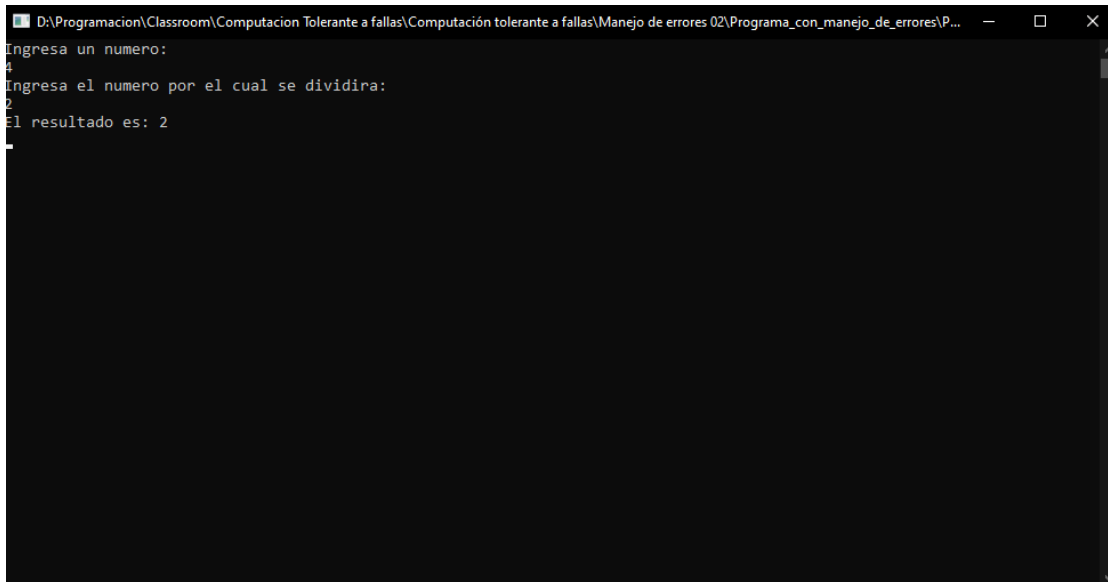


```
D:\Programacion\Classroom\Computacion Tolerante a fallas\Computación tolerante a fallas\Manejo de errores 02\Programa_con_manejo_de_errores\Programa_con_manejo_de_errores.exe
Ingresa un numero:
4
Ingresa el numero por el cual se dividira:
0
Error al momento de realizar la operacion
```

Con throw:



Corriendo con normalidad:

A screenshot of a Windows command prompt window. The title bar shows the file path: D:\Programacion\Classroom\Computacion Tolerante a fallas\Computación tolerante a fallas\Manejo de errores 02\Programa_con_manejo_de_errores\P... The console output shows the following text: 'Ingresa un numero:', '4', 'Ingresa el numero por el cual se dividira:', '2', and 'El resultado es: 2'. The user has entered '4' and '2' in response to the prompts.

Conclusión:

C# cuenta con algunas metodologías para la prevención de errores bastante útiles, sin embargo, como hemos visto en otra ocasión el hecho de poseer una metodología que busque prevenir en mayor medida la aparición de errores, no garantiza al 100% que estos nunca aparezcan, sin embargo como prevención son bastante útiles y permiten de cierta forma tener un programa lo más limpio posible y con garantía de los fallos no serán muy graves y se podrán solucionar.

Bibliografía:

- BillWagner. (2023, February 15). *Excepciones y control de excepciones*. Microsoft Learn. <https://learn.microsoft.com/es-es/dotnet/csharp/fundamentals/exceptions/>