

UNIVERSIDAD DE GUADALAJARA
Centro Universitario de Ciencias Exactas e Ingenierías

Computación Tolerante a Fallas



(Par. 2) Ejemplo: Otras herramientas para el manejar errores

Nombre: Castillo Mares Gilberto

Código: 213330514

Profesor: Dr. Michel Emanuel López Franco

2024-B

Índice

Contenido

| | |
|------------------|---|
| Desarrollo | 3 |
| Conclusión | 6 |

Desarrollo

Para el siguiente ejemplo utilicé el lenguaje de programación C++.

Código;

```
1  #include <iostream>
2  #include <stdexcept>
3
4  void dividir(int a, int b) {
5      if (b == 0) {
6          // Lanzar una excepción en lugar de usar
7          'raise'
8          throw std::runtime_error("Error:
Division por cero");
9      }
10     int resultado = a / b;
11     std::cout << "Resultado de la division: " <<
resultado << std::endl;
12 }
13
```

void dividir(int a, int b): Es una función que toma dos enteros, a y b, y realiza la división de a por b.

if (b == 0): Aquí se verifica si el divisor b es cero. Dividir entre cero es una operación indefinida, por lo que no debe permitirse.

throw std::runtime_error("Error: División por cero"); Si b es igual a 0, se lanza una excepción de tipo *std::runtime_error* con un mensaje de error. Esto interrumpe la ejecución normal de la función y transfiere el control al bloque catch correspondiente en la función que llamó a dividir.

int resultado = a / b: Si b no es cero, se realiza la división y se almacena el resultado en la variable resultado.

std::cout << "Resultado de la división: " << resultado << std::endl: Se imprime el resultado de la división.

```
14  int main() {
15      try {
16          dividir(10, 2);
17          dividir(8, 0); // Esto provocará una
excepción
18      } catch (const std::exception& e) {
19          std::cerr << "Excepcion capturada: " <<
e.what() << std::endl;
20      }
21
22      return 0;
23  }
```

int main(): Es la función principal del programa donde comienza la ejecución.

try: El bloque try contiene el código que puede lanzar una excepción.

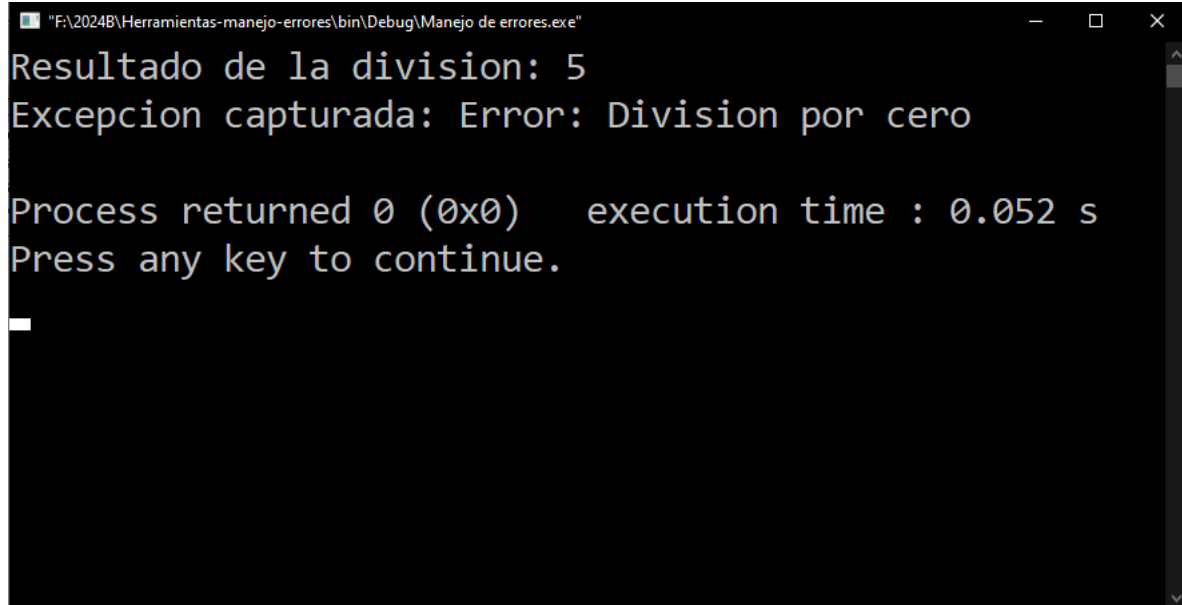
dividir(10, 2): Llama a la función dividir con los valores 10 y 2. Como 2 no es cero, la división se realiza correctamente y se imprime el resultado "5".

dividir(8, 0): Llama a la función dividir con los valores 8 y 0. Como el divisor es 0, la función dividir lanza una excepción, interrumpiendo su ejecución.

catch (const std::exception& e): Captura la excepción lanzada por dividir(8, 0) y maneja el error.

std::cerr << "Excepción capturada: " << e.what() << std::endl: Imprime un mensaje de error, indicando que una excepción fue capturada, junto con el mensaje de la excepción.

Consola:

A screenshot of a Windows command prompt window. The title bar shows the file path "F:\2024B\Herramientas-manejo-errores\bin\Debug\Manejo de errores.exe". The window has standard Windows window controls (minimize, maximize, close). The output text is as follows:

```
Resultado de la division: 5
Excepcion capturada: Error: Division por cero

Process returned 0 (0x0)   execution time : 0.052 s
Press any key to continue.
```

A small white cursor is visible on the line following the "Press any key to continue." message.

Al ejecutar el programa, la primera llamada a **dividir(10, 2)** imprimirá "Resultado de la división: 5".

La segunda llamada a **dividir(8, 0)** lanzará una excepción, que será capturada por el bloque catch, y se imprimirá "Excepción capturada: Error: División por cero".

Conclusión

Al utilizar la librería de excepciones ***stdexcept*** podemos utilizar algunas de sus funciones la cual utilicé en este ejemplo, la función utiliza mensajes de error los cuales son más fáciles de entender y ubicar rápido la ubicación del error, al utilizar la palabra reservada **throw** estamos manejando y señalando en tiempo de ejecución el error para después mostrarlo en pantalla si así se desea, una vez que la ejecución encuentra un problema, este mismo es dirigido al catch para continuar con la ejecución sin que el programa truene pero al mismo tiempo dando a conocer el tipo de error y porqué lo cual es un claro ejemplo de un programa tolerante a fallas, aunque es un ejemplo muy pequeño y puede haber muchísimos otros errores diferentes, para mi me queda claro lo que hice en este ejemplo.