



# 例外処理

応用編7日目

# 例外処理

- **SampleEx703**参照
- 例外とはプログラム動作中に起こる異常。
- 実行時エラーともいう。
- 理由は様々
  - 0による割り算が行われたため
  - 配列の範囲から外れてアクセスした

# 例外と例外処理

- 例外が発生したの処理 ... **例外処理**（れいがいしより）
- プログラム中に対処方法を記述できる
- try～catch～finallyで例外処理を記述することが出来る

# try/catchによる例外処理①

- try～catchは、{}内で例外が発生した場合、それに対処する処理を記述するためのもの

## **try～catchの書式**

```
try{  
    (処理①)  
}catch((例外クラス) 変数){  
    (処理②)  
}finally{  
    (処理③)  
}
```

# C#の代表的な例外クラス

例外クラス名	意味
<b>DivideByZeroException</b>	ゼロでの割り算をした
<b>IndexOutOfRangeException</b>	配列の範囲外のインデックスにアクセスした
<b>OverflowException</b>	算術演算・キャスト等の結果がオーバーフロー
<b>KeyNotFoundException</b>	コレクション内にアクセスするためのキーがない

# try/catchによる例外処理②

```
try
{
    for (int i = 0; i <= 5; i++)
    {
        int a = getNum(i);
        int b = 5;
        Console.Write(a + " / " + b + " = ");
        Console.WriteLine(calc(a, b));
    }
}
catch (DivideByZeroException e)
{
    Console.WriteLine();
    Console.WriteLine("0による割り算発生");
}
catch (IndexOutOfRangeException e)
{
    Console.WriteLine("配列の範囲外にアクセスしました");
}
finally
{
    Console.WriteLine("終了");
}
```

例外発生

# throw

- **throw** キーワードは例外を新たに発せさせることが出来るもの

throw キーワードの書式

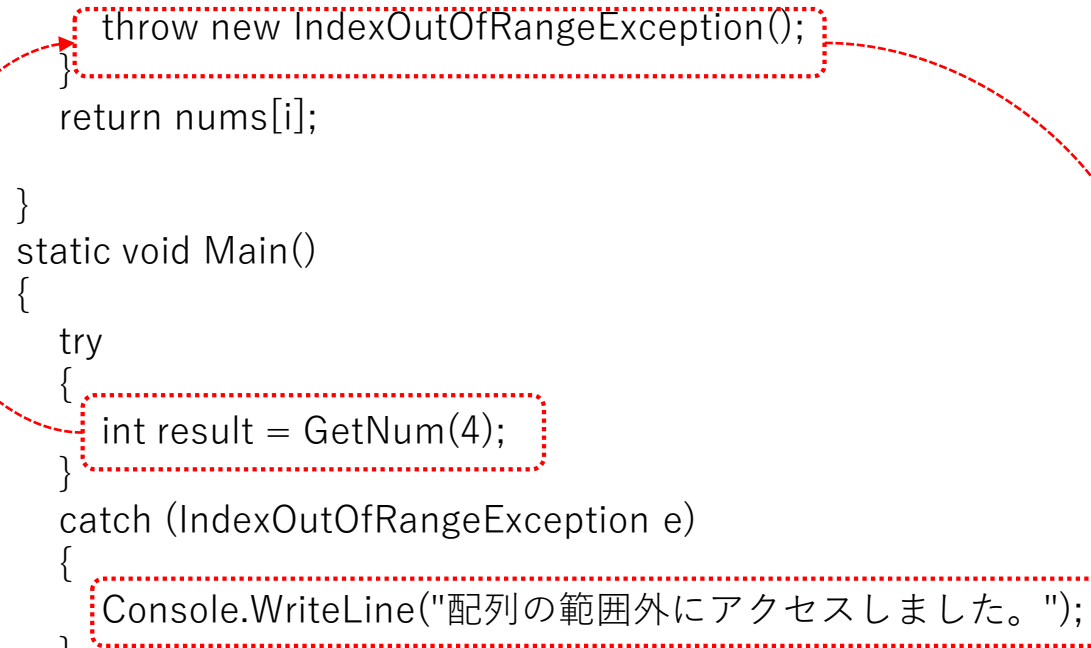
```
throw new 例外クラス()
```

- SampleEx704 参照

# throwによる例外の発生

```
static int GetNum(int i)
{
    int[] nums = { 300, 600, 900 };
    if (i > nums.Length)
    {
        // 例外を発生させる
        throw new IndexOutOfRangeException();
    }
    return nums[i];
}

static void Main()
{
    try
    {
        int result = GetNum(4);
    }
    catch (IndexOutOfRangeException e)
    {
        Console.WriteLine("配列の範囲外にアクセスしました。");
    }
}
```

A red dashed line with arrows illustrates the flow of an exception. It starts at the 'throw new IndexOutOfRangeException();' line in the GetNum method, loops around to the left, and then points to the 'Console.WriteLine' line in the catch block of the Main method.

例外発生