



# C# Basic



Проверить, идет ли запись

# Меня хорошо видно && слышно?



# Преподаватель



**Пилипсон Эдгар**

Senior Software Engineer, Luxoft



# Правила вебинара



Активно  
участвуем



Off-topic обсуждаем  
в Telegram  
**#csharp-basic-2024-05**



Задаем вопрос  
в чат или голосом



Вопросы вижу в чате,  
могу ответить не сразу

## Условные обозначения



Индивидуально



Время, необходимое  
на активность



Пишем в чат



Говорим голосом



Документ



Ответьте себе или  
задайте вопрос

# Исключения и их обработка

# Цели вебинара

После занятия вы сможете:

1. Изучить понятие исключения и их примеры
2. Использовать операторы работы с исключениями
3. Обращивать нестандартные случаи ошибок

# План полета

Что такое и как использовать

throw, try, catch, finally

Порядок перехвата

Условные исключения

Рефлексия



# Исключения (Exception)





# Что такое

**Исключение (exception)** – событие, означающее, что в программе что-то пошло не так (возникла ошибка)

Выбросить (пробросить) исключение – вызвать событие исключения

- Поделили на ноль
- Аргумент функции некорректный
- Доступ к полю неинициализированного объекта
- Выход за границы массива

# Когда могут появляться

- Во время выполнения .NET Runtime (*доступ к пустому объекту*)
- Выбрасываются сторонними библиотеками
- Вызываются пользователем принудительно

# Как выглядят

```
var a = 0;  
var b = 4;  
Console.WriteLine(b / a);
```

```
Unhandled exception. System.DivideByZeroException: Attempted to divide by zero.  
   at Otus.Exceptions.Program.Main(String[] args) in C:\Users\Эдгар\Documents\otus\modules\csharp-base\8 - Исключения  
и их обработка\code\Otus.Exceptions\Otus.Exceptions\Program.cs:line 11
```

# Немного про классы

# Что такое

Тип для описания какого-то сложного объекта, с разными функциями, полями и пр.

```
// Это класс под названием
// Транспортное средство
class Vehicle
{
    // у него есть свойство Название
    public string Name;
}
```

# System.Exception

# System.Exception

**System.Exception** – базовый класс исключений

Все исключения наследуются на определенном уровне от System.Exception

Принято все классы исключений называть с суффиксом Exception

- InvalidOperation**Exception**
- OutOfRange**Exception**
- ArgumentNullException**Exception**

# Собственные исключения

```
/// <summary>  
/// Мое собственное исключение  
/// </summary>  
class MyCystomException : Exception  
{  
  
}
```



# Свойства

- Message – сообщение в исключении
- StackTrace – текстовая информация с порядком вызова места исключения
- InnerException – исключение, вызвавшее данное (если есть)
- Data – словарь ключ-значение с доп. данными (например, параметрами функции)
- TargetSite – информация о методе, где произошла ошибка

# Операторы

# Общее, синтаксис

```
try
{
    if (b == 0)
        throw new DivideByZeroException("Делим на ноль");
    return a / b;
}
catch (Exception e)
{
    Console.WriteLine("Произошла ошибка");
    return 0.0;
}
finally
{
    Console.WriteLine("Я блок finally ");
}
```

# throw

```
try
{
    if (b == 0)
        throw new DivideByZeroException("Делим на ноль");
    return a / b;
}
catch (Exception e)
{
    Console.WriteLine("Произошла ошибка");
    return 0.0;
}
finally
{
    Console.WriteLine("Я блок finally ");
}
```



# throw

**throw** – оператор вызова исключения

Синтаксис

**throw** *объект\_класс\_exception\_или\_производного*

# throw

Можно так

```
throw new Exception("Я сообщение об ошибке");
```

А можно так

```
var ex = new Exception("Я сообщение об ошибке");  
ex.Data.Add("a", "2");  
throw ex;
```

# try catch

```
try
{
    if (b == 0)
        throw new DivideByZeroException("Делим на ноль");
    return a / b;
}
catch (Exception e)
{
    Console.WriteLine("Произошла ошибка");
    return 0.0;
}
finally
{
    Console.WriteLine("Я блок finally ");
}
```

# try catch

**try catch** – операторы перехвата исключений

**try {}** – объявляет область кода, где потенциально может вызываться исключение

**catch{}** – «ловит» исключение из блока try, где можно его обработать



# try catch

```
try
{
    // Здесь может быть какой-то код
}
catch (Exception exc)
{
    // А здесь происходит обработка исключения exc
    // вызванного в каком-то коде
}
```

# try catch

```
try
{
    // Здесь может быть какой-то код
}
catch (FooException exc)
{
    // Если какой-то код выкинул FooException
}
catch (BarException exc)
{
    // Если какой-то код выкинул BarException
}
```

# try catch

```
try
{
    // Здесь может быть какой-то код
}
catch (Exception exc)
{
    ОбработкаИсключения(exc);

    // пробрасываем то же исключение
    throw;
}
```

# finally

```
try
{
    if (b == 0)
        throw new DivideByZeroException("Делим на ноль");
    return a / b;
}
catch (Exception e)
{
    Console.WriteLine("Произошла ошибка");
    return 0.0;
}
finally
{
    Console.WriteLine("Я блок finally ");
}
```

# finally

**finally** – оператор выполнения кода после всех обработок

- Блок внутри выполняется всегда, есть исключение или нет
- в блоке нельзя возвращать что-то (*return*)

# finally

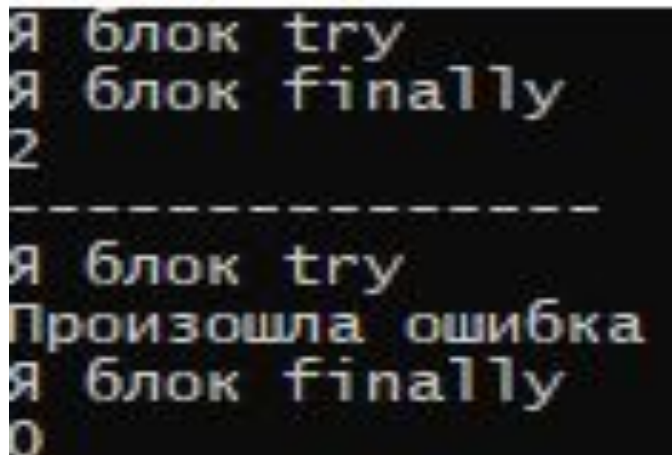
**finally** – оператор выполнения кода после всех обработок

- Блок внутри выполняется всегда, есть исключение или нет
- в блоке нельзя возвращать что-то (*return*)

# finally

```
double Divide(int a, int b)
{
    try
    {
        Console.WriteLine("Я блок try");
        return a / b;
    }
    catch (Exception)
    {
        Console.WriteLine("Произошла
ошибка");
        return 0;
    }
    finally
    {
        Console.WriteLine("Я блок finally");
    }
}
```

```
Console.WriteLine(Divide(4,2));
Console.WriteLine("-----");
Console.WriteLine(Divide(4, 0));
```



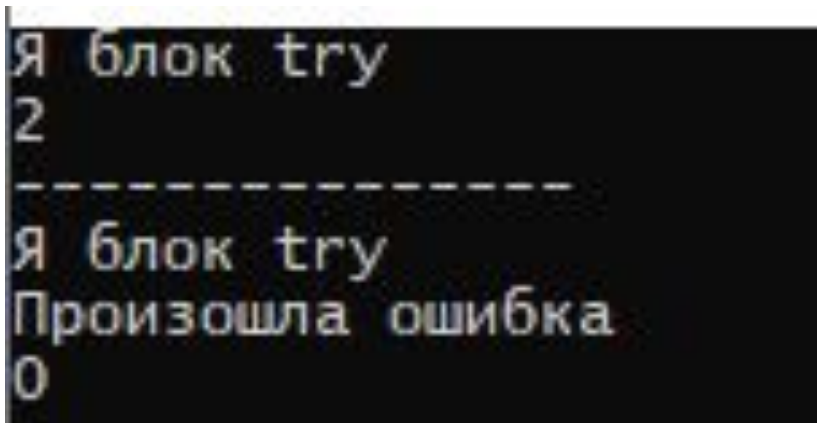
```
Я блок try
Я блок finally
2
-----
Я блок try
Произошла ошибка
Я блок finally
0
```

# finally

```
double Divide(int a, int b)
{
    try
    {
        Console.WriteLine("Я блок try");
        return a / b;
    }
    catch (Exception)
    {
        Console.WriteLine("Произошла
ошибка");
        return 0;
    }

    Console.WriteLine("Я блок finally");
}
```

```
Console.WriteLine(Divide(4,2));
Console.WriteLine("-----");
Console.WriteLine(Divide(4, 0));
```



```
Я блок try
2
-----
Я блок try
Произошла ошибка
0
```



# Порядок перехвата исключений

# Порядок перехвата исключений

// Исключение-болезнь

```
class IllnessException : Exception { }
```

// Микробная болезнь

```
class MicrobeException : IllnessException { }
```

// Вирусная болезнь

```
class VirusException : IllnessException { }
```

# Порядок перехвата исключений

```
static void DemoCure()
{
    try
    {
        Live();
    }
    catch (VirusException)
    {
        // тут ловим ТОЛЬКО VirusException
    }
    catch (IllnessException)
    {
        // тут ловим IllnessException и производные,
        // в т.ч. MicrobeException
        // НО НЕ VirusException
    }
    catch (Exception)
    {
        // тут ловим все остальные исключения
    }
}
```



# Перехват на разных уровнях функций

```
class RedException : Exception
{
    public RedException() : base("КОД КРАСНЫЙ")
    {
    }

    public RedException(string message) :
base(message)
    {
    }
}
```

```
class PurpleException :
RedException
{
    public PurpleException() :
base("КОД ФИОЛЕТОВЫЙ")
    {
    }
}
```



# Перехват на разных уровнях функций

```
void Throw(WhatToThrow a)
{
    switch (a)
    {
        case WhatToThrow.Red:
            throw new RedException();
        case WhatToThrow.Purple:
            throw new PurpleException();
        default:
            throw new InvalidOperationException($"Непонятная ошибка");
    }
}
```

# Перехват на разных уровнях функций

```
static void Level2(WhatToThrow a)
{
    try
    {
        Throw(a);
    }
    catch (PurpleException e)
    {
        Console.Write($"LEVEL2: ФИОЛЕТОВОЕ '{e.Message}'");
    }
    catch (RedException e)
    {
        Console.Write($"LEVEL2: КРАСНОЕ '{e.Message}'");
        throw;
    }
}
```

```
static void Level1(WhatToThrow a)
{
    try
    {
        Level2(a);
    }
    catch (Exception e)
    {
        Console.WriteLine($"Level1: Я жду любую ошибку '{e.Message}'");
    }
}
```

# Дополнительные техники

# Упрощаем catch

```
try
{
    // Тут какой-то код
}
catch (ArgumentNullException exception)
{
    // если в блоке с объектом
    // exception не работаем,
    // но важно перехватить именно
    // ArgumentNullException
    // exception можно убрать
}
```

=

```
try
{
    // Тут какой-то код
}
catch (ArgumentNullException)
{
    // на важен сам факт возникновения
    // ArgumentNullException
}
```





# Упрощаем catch

```
try
{
    // Тут какой-то код
}
catch (ArgumentNullException exception)
{
    // если в блоке с объектом
    // exception не работаем,
    // но важно перехватить именно
    // ArgumentNullException
    // exception можно убрать
}
```

=

```
try
{
    // Тут какой-то код
}
catch (ArgumentNullException)
{
    // на важен сам факт возникновения
    // ArgumentNullException
}
```



# Упрощаем catch - 2

```
try
{
    // Тут какой-то код
}
catch (Exception)
{
    // Если в блоке ловим
    // любые исключения
    // и нам не нужно их содержание
    // Exception убираем
}
```

=

```
try
{
    // Тут какой-то код
}
catch
{
    // обрабатываем любые исключения
}
```



# Условные исключения

```
int GetItem(int[] arr, int index)
{
    try
    {
        return arr[index];
    }
    catch (IndexOutOfRangeException) when (index < 0) // IndexOutOfRangeException для index
    < 0
    {
        Console.WriteLine("Индекс меньше нуля");
    }
    catch (IndexOutOfRangeException) // IndexOutOfRangeException в остальных случаях
    {
        Console.WriteLine("Индекс аут оф рэндж");
    }
    catch // остальные ошибки
    {
        Console.WriteLine("Другая ошибка");
    }
    return 0;
}
```



# Выводы и рефлексия

# Цели вебинара

## Проверка достижения целей

1. Исключения - удобный механизм обработки ситуаций, когда что-то пошло не так
2. В C# - богатый инструментарий работы с исключениями

# Рефлексия



По какому вопросу захотелось глубже изучить информацию?



Понимаете ли вы, как применять на практике то, что узнали на вебинаре. Если да, то как?

# Следующий вебинар



5 июля 2024

Модуль 1: Знакомство с C#

## Групповая менторская консультация



Ссылка на вебинар  
будет в ЛК за 15 минут



Материалы  
к занятию в ЛК —  
можно изучать



Обязательный материал  
обозначен красной  
лентой



**Заполните, пожалуйста,  
опрос**



**Спасибо за внимание!**  
**Приходите на следующие**  
**вебинары**