Информатика

Фоменко Максим Юрьевич

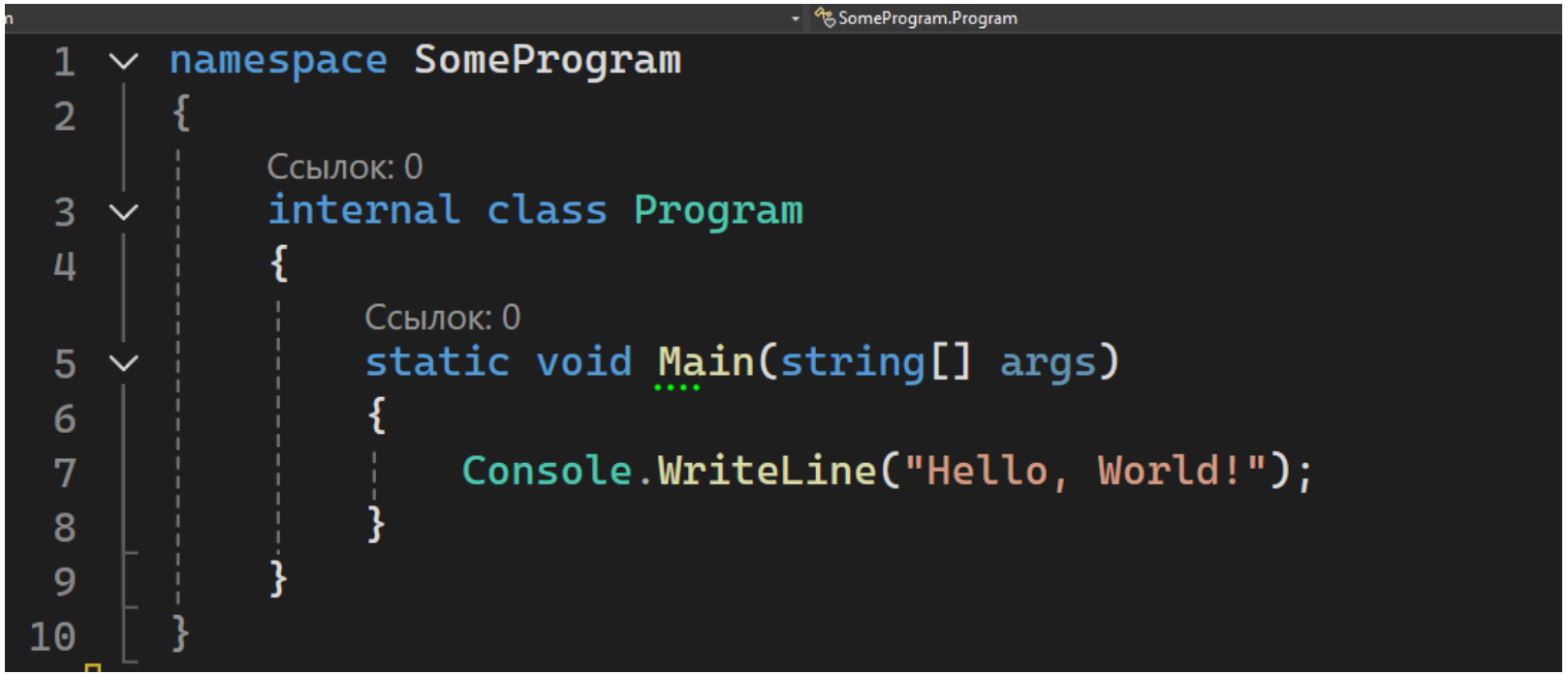
Про что курс?

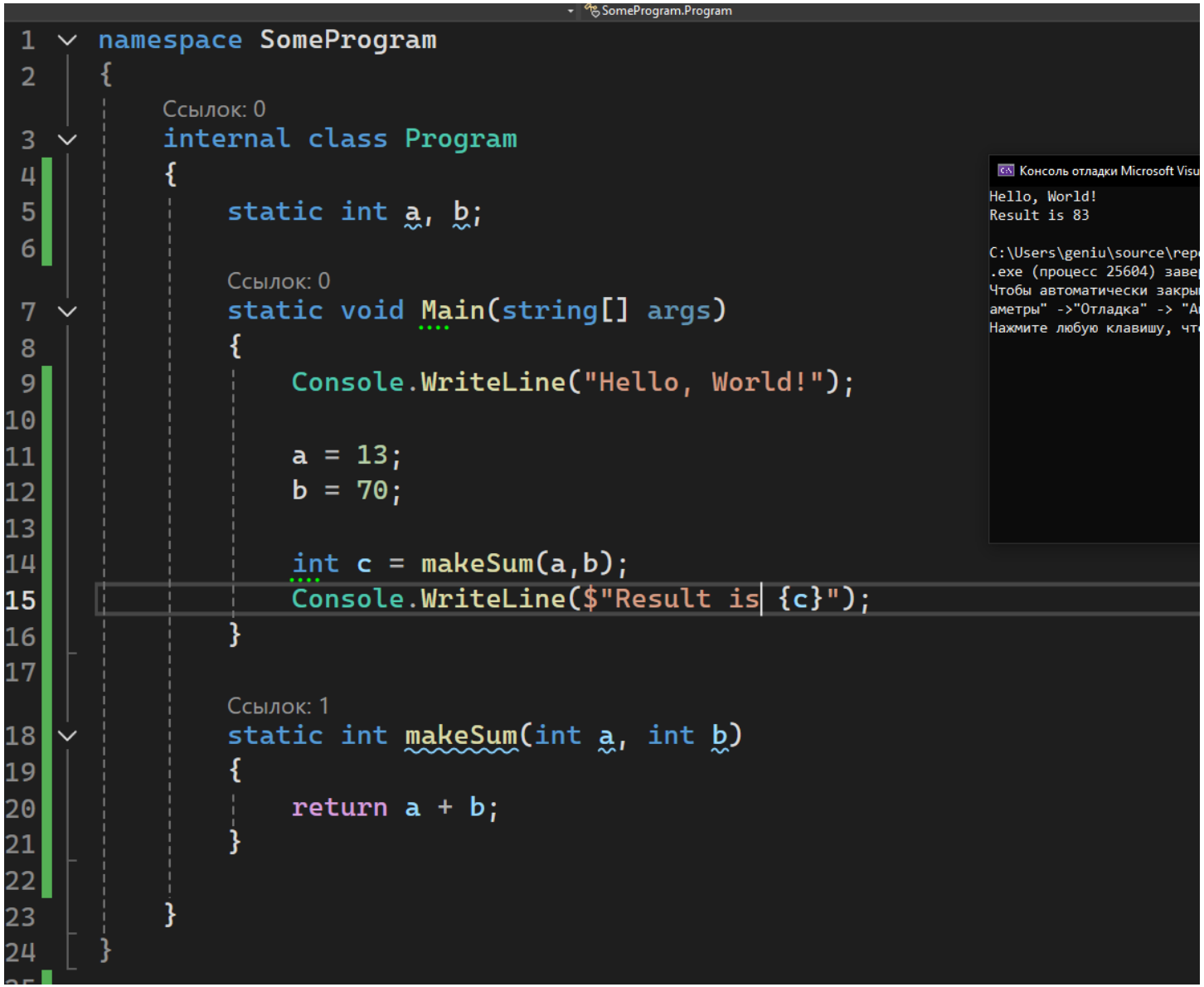
- Переменные, арифметические операции, циклы, условия
- Массивы и строки
- Функции
- Обработка исключений и работа с файлами в C#
- Графические приложения на Windows Presentation Foundation

Продолжение

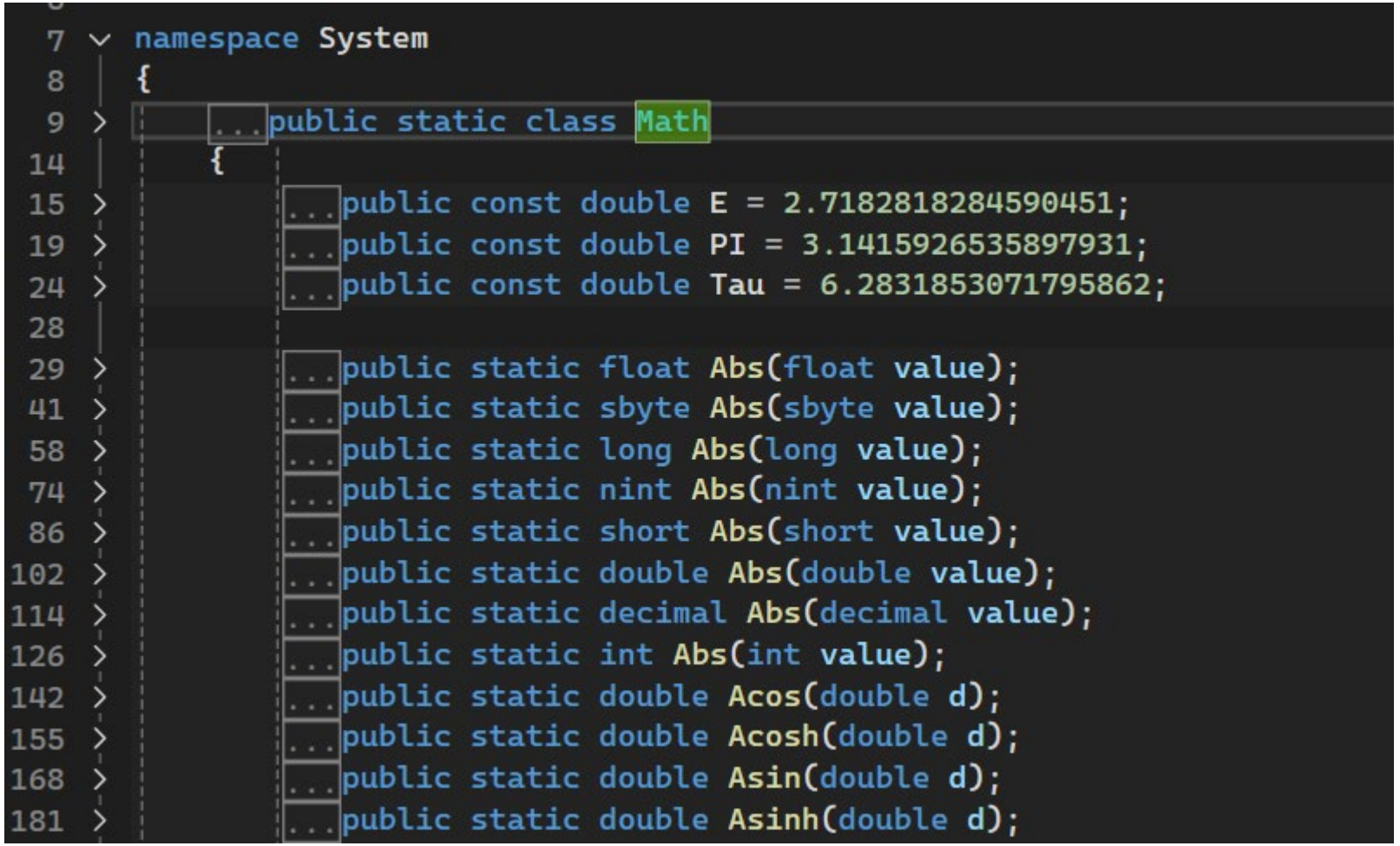
- 1 семестр: информатика
- 2 семестр: программирование и обработка графических интерфейсов 1
- 3 семестр: программирование и обработка графических интерфейсов 2
- Разработка игр, разработка мобильных приложений, вебдиз

Структура программы




SomeProgram.Program
namespace SomeProgram
{
Ссылок: 0
internal class Program
{
static int a, b;
Ссылок: 0
static void Main(string[] args)
{
Console.WriteLine("Hello, World!");
a = 13;
b = 70;
int c = makeSum(a,b);
Console.WriteLine($"Result is {c}");
}
Ссылок: 1
static int makeSum(int a, int b)
{
return a + b;
}
}
}
Консоль отладки Microsoft Visu
Hello, World!
Result is 83
C:\Users\geniu\source\rep
.exe (процесс 25604) заве
Чтобы автоматически закры
аметры" ->"Отладка" -> "А
Нажмите любую клавишу, чт

Класс Math


namespace System
{
public static class Math
{
public const double E = 2.7182818284590451;
public const double PI = 3.1415926535897931;
public const double Tau = 6.2831853071795862;
public static float Abs(float value);
public static sbyte Abs(sbyte value);
public static long Abs(long value);
public static nint Abs(nint value);
public static short Abs(short value);
public static double Abs(double value);
public static decimal Abs(decimal value);
public static int Abs(int value);
public static double Acos(double d);
public static double Acosh(double d);
public static double Asin(double d);
public static double Asinh(double d);

static

```
a = 13;
```

(Поле) int Program.a

CS0120: Для нестатического поля, метода или свойства "Program.a" требуется ссылка на объект.

Cannot access non-static field 'a' in static context

```
Ссылок: 0
internal class Program
{
    int a, b;

    Ссылок: 0
    static void Main(string[] args)
    {
        Console.WriteLine("Hello, World!");

        a = 13;
    }

    Ссылок: 0
    void Test()
    {
        a = 13;
        b = 152;
    }

    Ссылок: 0
    static int makeSum(int a, int b)
    {
        return a + b;
    }

}
```

Переменные и типы данных

Переменная - именованная область памяти, к которой можно обращаться за данными, используя идентификатор (имя переменной).

Данные, находящиеся в переменной (то есть по её адресу в памяти), называются значением этой переменной.

В C# все типы данных делятся на две большие категории: **значимые типы (Value Types)** и **ссылочные типы (Reference Types)**

Переменные и типы данных

Переменная значимого типа напрямую содержит свое **значение**.

- значимые: int, double, bool, char

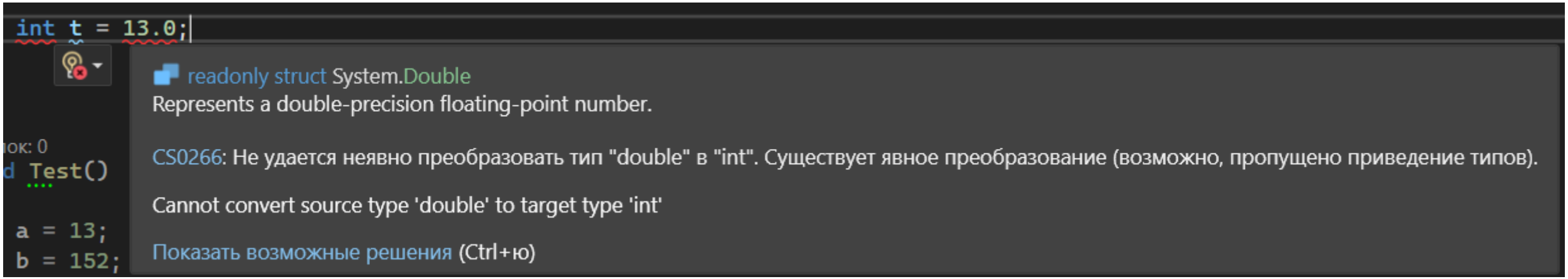
Переменная ссылочного типа хранит не сам объект, а **ссылку (или адрес)** на место в памяти, где этот объект находится.

- Ссылочные: string, все массивы (int[], string[]), все классы (class), object, делегаты.

Переменные и типы данных

C# строго типизированный язык. Это означает, что **компилятор проверяет код на ошибки, связанные с типами, ещё до его выполнения**.

- Пример: в питоне или JS вы получите ошибку связанную с типом только во время итерации программы, в C# до запуска IDE нас предупредит.



Логический и символьный типы

Тип в C#	**Тип в .NET**	**Размер**	**Значения**
`bool`	`System.Boolean`	1 байт	`true` или `false`
`char`	`System.Char`	2 байта (16 бит)	Один символ Unicode (например, 'A', 'щ', '$')

Целочисленные типы со знаком (Signed Integers)

Тип в C#	**Тип в .NET**	**Размер**	**Диапазон значений (от min до max)**
`sbyte`	`System.SByte`	1 байт (8 бит)	от -128 до 127
`short`	`System.Int16`	2 байта (16 бит)	от -32 768 до 32 767
`int`	`System.Int32`	4 байта (32 бита)	от -2 147 483 648 до 2 147 483 647
`long`	`System.Int64`	8 байт (64 бита)	от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807

Целочисленные типы без знака (Unsigned Integers)

Тип в C#	**Тип в .NET**	**Размер**	**Диапазон значений (от min до max)**
`byte`	`System.Byte`	1 байт (8 бит)	от 0 до 255
`ushort`	`System.UInt16`	2 байта (16 бит)	от 0 до 65 535
`uint`	`System.UInt32`	4 байта (32 бита)	от 0 до 4 294 967 295
`ulong`	`System.UInt64`	8 байт (64 бита)	от 0 до 18 446 744 073 709 551 615

Типы с плавающей точкой (Floating-Point Types)

Тип в C#	**Тип в .NET**	**Размер**	**Точность**	**Приблизительный диапазон**
`float`	`System.Single`	4 байта (32 бита)	~6-9 знаков	от $\pm 1.5 \times 10^{-45}$ до $\pm 3.4 \times 10^{38}$
`double`	`System.Double`	8 байт (64 бита)	~15-17 знаков	от $\pm 5.0 \times 10^{-324}$ до $\pm 1.7 \times 10^{308}$

Тип высокой точности (High-Precision Type)

Тип в C#	**Тип в .NET**	**Размер**	**Точность**	**Приблизительный диапазон**
`decimal`	`System.Decimal`	16 байт (128 бит)	28-29 знаков	от $\pm 1.0 \times 10^{-28}$ до $\pm 7.928 \times 10^{28}$

Ссылочные типы (Reference Types)

Тип в C#	**Тип в .NET**	**Размер**	**Описание**
`string`	`System.String`	Зависит от длины	Последовательность символов Unicode.
`object`	`System.Object`	Размер ссылки (4 или 8 байт)	Базовый тип для всех остальных типов в .NET. Переменная `object` м ожет ссылаться на экземпляр любого типа.

Переменные

```
Ссылок: 0
static void Main(string[] args)
{
    int a; // Объявили
    a = 10; // инициализировали

    int b = 20; //Объявили и проинициализировали

}
```

```
Ссылок: 0
static void Main(string[] args)
{
    string name = "Tom";
    bool isEmployed = false;
    double weight = 78.65;

}
```

```
Ссылок: 0
static void Main(string[] args)
{
    var a = 15;
    var b = $"Test string {a}";

}
```

Математические операции

- Сложение (+): int a = 10 + 5; // a будет равно 15
- Вычитание (-): int b = 10 - 5; // b будет равно 5
- Умножение (): int c = 10 5; // c будет равно 50
- Деление (/) : int d = 10 / 5; // d будет равно 2

Нюанс при делении

Если вы делите два целых числа, результат также будет целым, а дробная часть отбрасывается.

Например:

- int a = 9 / 5 //Результат будет 1, так как оба операнда целые числа
- double a = 9.0 / 5 //Результат будет 1.8.

Остаток

Оператор (%) возвращает остаток от целочисленного деления.

int e = 10 % 3; // e будет равно 1 (10 делим на 3, в остатке 1)

Инкремент и декремент

(++, --): Эти унарные операторы увеличивают или уменьшают значение переменной на единицу.

```
Ссылок: 0
static void Main(string[] args)
{
    int f = 5;
    f++; // f теперь равно 6
    int g = 5;
    g--; // g теперь равно 4

}
```

Класс Math

Для более сложных математических вычислений, таких как возведение в степень, извлечение квадратного корня или тригонометрические функции (синус, косинус), в C# существует встроенный класс **Math**.

Логические операции

Логические операции используются для работы с булевым типом данных (bool), который может принимать только два значения: true (истина) или false (ложь).

Они являются основой для создания условий и управления потоком выполнения программы.

Логические операции

Логическое "И" (&& - Амперсанд): Выражение a && b истинно (true) только в том случае, если и a, и b истинны.

```
Ссылок: 0
static void Main(string[] args)
{
    bool result = (5 > 3) && (10 > 5); // result будет true, так как оба условия верны.

}
```

Логические операции

Логическое "ИЛИ" (|| - пайп): Выражение a || b истинно (true), если хотя бы одно из выражений, a или b, истинно.

```
Ссылок: 0
static void Main(string[] args)
{
    bool result = (5 < 3) || (10 > 5); // result будет true, так как второе условие верно.
}
```

Логические операции

Логическое "НЕ" (!): Этот унарный (работает только с одним аргументом) оператор инвертирует (меняет на противоположное) логическое значение.

```
Ссылок: 0
static void Main(string[] args)
{
    bool result = !(5 > 3);  // result будет false,
                             // так как (5 > 3) - это true,
                             // а "НЕ" true - это false.
}
```

Операторы сравнения

Логические операторы используются вместе с операторами сравнения, которые сравнивают два значения и возвращают результат в виде bool.

- == (равно)
- != (не равно)
- < (меньше)
- > (больше)
- <= (меньше или равно)
- >= (больше или равно)

Массивы

Массив (array) — структура данных в компьютерном программировании и информатике. Это упорядоченный набор однотипных элементов, хранящихся в смежных ячейках памяти.

Массив в C# - это ссылочный тип данных, представляющий собой индексированную коллекцию элементов одного и того же типа, размещенных в непрерывной области памяти.

Массивы

Ключевые характеристики массива:

1. Массив может содержать элементы только одного, строго определенного при его создании, типа (например, int[] для целых чисел или string[] для строк).
2. Размер массива определяется в момент его создания (инициализации) и не может быть изменен в дальнейшем.
3. Доступ к элементам массива осуществляется по целочисленному индексу. Нумерация индексов всегда начинается с нуля. Первый элемент имеет индекс 0, второй - 1, и так далее, до Length - 1.

Объявление и инициализация массива

Декларация

На этом этапе мы объявляем переменную, которая будет ссылаться на объект массива в памяти.

```
Ссылок: 0
static void Main(string[] args)
{
    // Декларация переменной 'numbers', которая может ссылаться на массив целых чисел.
    int[] numbers;
}
```

Объявление и инициализация массива

Инициализация

Инициализация - процесс создания экземпляра массива с помощью оператора **new**. В этот момент мы обязаны указать его размер.

```
Ссылок: 0
static void Main(string[] args)
{
    int[] numbers;

    numbers = new int[5];
}
```

Объявление и инициализация массива

```
Ссылок: 0
static void Main(string[] args)
{
    int[] numbers = new int[5];
}
```

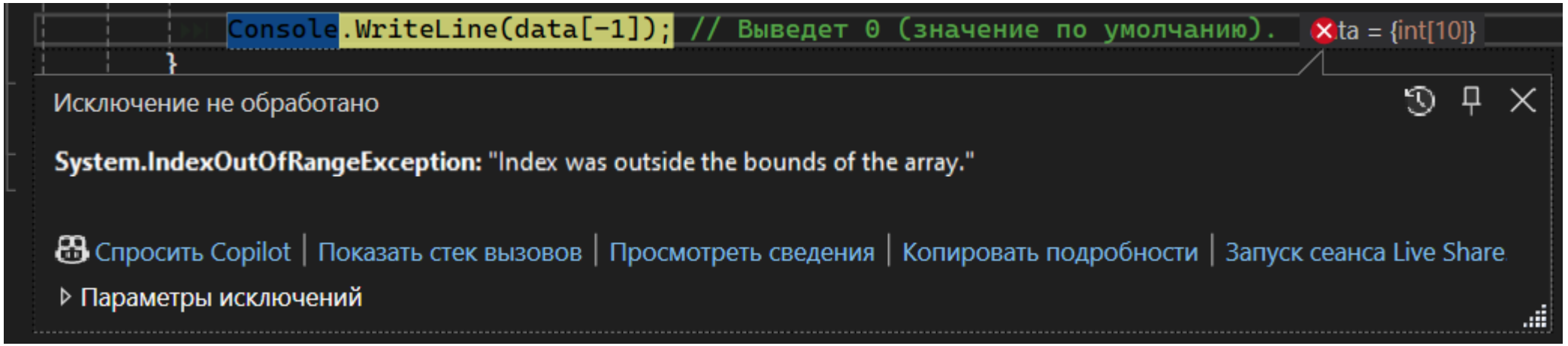
Доступ к элементам массива

Доступ для чтения или записи элемента осуществляется через указание имени массива и индекса элемента в квадратных скобках [].

```
Ссылок: 0
static void Main(string[] args)
{
    int[] data = new int[10];

    // Запись значения в элемент
    // с индексом 3 (четвертый по счету элемент).
    data[3] = 99;

    // Чтение значения из элемента с индексом 3.
    int value = data[3]; // value будет равно 99

    Console.WriteLine(data[0]); // Выведет 0 (значение по умолчанию).
}
```

Доступ к элементам массива

Попытка обратиться к элементу по индексу, который находится за пределами допустимого диапазона (т.е. меньше 0 или больше либо равен Length), приведет к генерации исключения System.IndexOutOfRangeException во время выполнения программы.



Свойство Length и итерация по массиву

Каждый массив имеет свойство Length, которое возвращает общее количество элементов, которое может содержать массив.

```
Ссылок: 0
static void Main(string[] args)
{
    double[] measurements = new double[250];
    Console.WriteLine(measurements.Length); // Выведет 250
}
```

Цикл for

Классический способ итерации, который предоставляет полный контроль над процессом, включая доступ к индексу элемента.

```
Ссылок: 0
static void Main(string[] args)
{
    double[] measurements = new double[250];
    Console.WriteLine(measurements.Length); // Выведет 250

    for (int i = 0; i < measurements.Length; i++)
    {
        // i будет последовательно принимать значения от 0 до 249.
        measurements[i] = i * 1.5; // Пример операции с элементом.
    }
}
```

Многомерные массивы

C# поддерживает многомерные массивы, которые полезны для представления табличных данных, матриц или сеток.

Массивы характеризуются таким понятием как ранг или количество измерений. Выше мы рассматривали массивы, которые имеют одно измерение (то есть их ранг равен 1) - такие массивы можно представлять в виде ряда (строки или столбца) элемента. Но массивы также бывают многомерными. У таких массивов количество измерений (то есть ранг) больше 1.

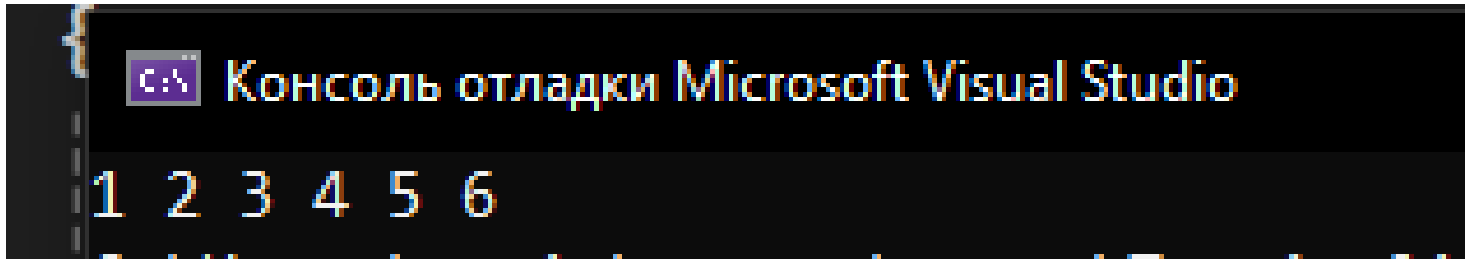
Многомерные массивы

```
// Декларация и инициализация двумерного массива (матрицы 3x4).
int[,] matrix = new int[3, 4];

// Доступ к элементу на пересечении строки с индексом 1 и столбца с индексом 2.
matrix[1, 2] = 5;
```

Многомерные массивы

Определенную сложность может представлять перебор многомерного массива. Прежде всего надо учитывать, что длина такого массива - это совокупное количество элементов.

```
int[,] numbers = { { 1, 2, 3 }, { 4, 5, 6 } };
foreach (int i in numbers)
    Console.Write($"{i} ");
```



Многомерные массивы

Для получения размера каждого измерения используется метод GetLength(dimension).

```
int rows = numbers.GetLength(0);    // Вернет 3
int cols = numbers.GetLength(1);    // Вернет 4
```