Лекция №2
по курсу
«Системное программирование»

Лектор: д.т.н., Оцоков Шамиль Алиевич,

email: otsokovShA@mpei.ru

Москва, 2021

Классификация типов переменных

Операции

Таблица 3.1. Операции C#

Категория	Знак операции	Название
Первичные	x()	Вызов метода или делегата
	x[]	Доступ к элементу
	x++	Постфиксный инкремент
	x--	Постфиксный декремент
	new	Выделение памяти
Унарные	+	Унарный плюс
	-	Унарный минус (арифметическое отрицание)
	!	Логическое отрицание
	~	Поразрядное отрицание
	++x	Префиксный инкремент
	--x	Префиксный декремент
	(тип)x	Преобразование типа
Мультипликативные (типа умножения)	*	Умножение
	/	Деление
	%	Остаток от деления
Аддитивные (типа сложения)	+	Сложение
	-	Вычитание

Операции

Сдвига	<<	Сдвиг влево
	>>	Сдвиг вправо
Отношения и проверки типа	<	Меньше
	>	Больше
	<=	Меньше или равно
	>=	Больше или равно
	is	Проверка принадлежности типу
	as	Приведение типа
Проверки на равенство	==	Равно
	!=	Не равно

Поразрядные логические	&	Поразрядная конъюнкция (И)
	^	Поразрядное исключающее ИЛИ
	|	Поразрядная дизъюнкция (ИЛИ)
Условные логические	&&	Логическое И
	||	Логическое ИЛИ
Условная	? :	Условная операция
Присваивания	=	Присваивание
	*=	Умножение с присваиванием
	/=	Деление с присваиванием
	%=	Остаток от деления с присваиванием
	+=	Сложение с присваиванием

Операции

+=	Сложение с присваиванием
-=	Вычитание с присваиванием
<<=	Сдвиг влево с присваиванием
>>=	Сдвиг вправо с присваиванием
&=	Поразрядное И с присваиванием
^=	Поразрядное исключающее ИЛИ с присваиванием
|=	Поразрядное ИЛИ с присваиванием

Формат с фиксированной точкой

xxxxx.yyyyy

0,12377

12345,45454

3334,123

Формат с плавающей точкой

$$x = \pm S \cdot 10^{E}, \quad 1 \le S < 10$$

$$x = \pm S \cdot 2^{E}, \quad 1 \le S < 2$$

$0{,}123 = 0{,}123 \cdot 10^{0}$

$0{,}123 = 123 \cdot 10^{-3}$

$0{,}123 = 1.23 \cdot 10^{-1}$

$X = 0{,}123 * 10^{0}$

$Y = 0{,}1 * 10^{2}$

$X * Y = 0{,}123 * 0{,}1 * 10^{2+0}$

X+Y = ?

$0{,}123 * 10^{0} + 0{,}1 * 10^{2}$

$= 0{,}00123 * 10^{2} + 0{,}1 * 10^{2}$

$= 0{,}10123 * 10^{2}$

Преобразование типов

```
double  y = 4.12;
decimal d = 600m;
string  s = "Вася";

Console.WriteLine( "i = " + i );                    // 1
Console.WriteLine( "s = " + s );                    // 2
Console.WriteLine( "y = {0} \nd = {1}", y, d );     // 3
```

```
// Declare a variable
char ch = 'a';
// Print the results on the console
Console.WriteLine(
   "The code of '" + ch + "' is: " + (int)ch);
```

Вывод строки

```
string path = "C:\\Windows\\Notepad.exe";

Console.WriteLine(path);

string verbatim = @"The \ is not escaped as \\. I am at a new line.";

Console.WriteLine(verbatim);
```

Условный оператор

```
If (условие)
{
   тело условия
}
Elseif …
Else ..
```

&& - логическое И

|| - логическое или

Циклы

a++ инкремент => a = a + 1

a-- декремент => a = a -1

a +=5 => a = a + 5

a -=5 => a = a -5

Цикл while (условие)

```
public class Loop {
    public static void main(String[] args) {
        int i = 1;
        while (i <= 1000) {
            System.out.println(i);
            i++;
        }
    }
}
```

break – прерывание цикла

```
public class Loop {
    public static void main(String[] args) {
        int i = 1;
        while (i <= 1000){
            System.out.println(i);
            i++;
            if (i == 5) {
                break;
            }
        }
    }
}
```

```
public class Loop {
    public static void main(String[] args) {
        int sum = 0;
        int count = 1;
        while (count <= 100) {
            sum += count;
            count++;
        }
        float result = sum / (float) count;
        System.out.println(result);
    }
}
```

МАССИВЫ

№	Объявление массива, Java-синтаксис	Примеры	Комментарий

1.	`dataType[] arrayName;`	`int[] myArray;`
		
		`Object[] arrayOfObjects;`
			Желательно объявлять массив именно таким способом, это Java-стиль

Как и любой другой объект, создать массив Java, то есть зарезервировать под него место в памяти, можно с помощью оператора **`new`**. Делается это так:

```
1  new typeOfArray [length];
```

Где **`typeOfArray`** — это тип массива, а **`length`** — его длина (то есть, количество ячеек), выраженная в целых числах (`int`). Однако здесь мы только выделили память под массив, но не связали созданный массив ни с какой объявленной ранее переменной. Обычно массив сначала объявляют, а потом создают, например:

```
1  int[] myArray; // объявление массива
2  myArray = new int[10]; // создание, то есть, выделение памяти для массива на 10 элементов типа int
```

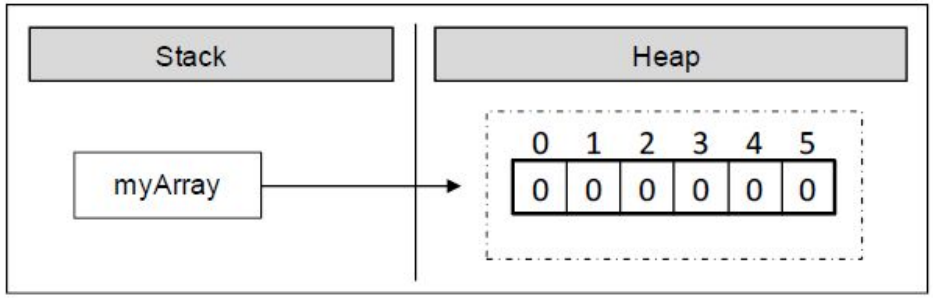
МАССИВЫ

Creation of an Array – the Operator "new"

In C# we create an array with the help of the keyword **new**, which is used to allocate memory:

```
int[] myArray = new int[6];
```

In this example we allocate an array with length of 6 elements of type **int**. This means that in the dynamic memory (heap) an area of 6 integer numbers is allocated and they all are initialized with the value 0:



МАССИВЫ

```
int[] array = new int[] { 1, 2, 3, 4, 5 };

Console.Write("Output: ");
for (int index = 0; index < array.Length; index++)
{
   // Doubling the number
   array[index] = 2 * array[index];
   // Print the number
   Console.Write(array[index] + " ");
}
// Output: 2 4 6 8 10
```

МАССИВЫ

```
int[] arr = new int[5];
for (int i = 0; i < arr.Length; i++)
{
   arr[i] = i;
}
```

```
int n = int.Parse(Console.ReadLine());
int[] array = new int[n];
```

```
for (int i = 0; i < n; i++)
{
   array[i] = int.Parse(Console.ReadLine());
}
```

```
string[] array = { "one", "two", "three", "four" };

for (int index = 0; index < array.Length; index++)
{
   // Print each element on a separate line
   Console.WriteLine("Element[{0}] = {1}", index, array[index]);
}
```

МАССИВЫ

```
string[] capitals =
   { "Sofia", "Washington", "London", "Paris" };

foreach (string capital in capitals)
{
   Console.WriteLine(capital);
}
```

МАССИВЫ

int[] myArray = new int[10];
получаем массив из десяти целых чисел, и, пока это не измениться в ходе программы, в каждой ячейке записан 0. массив с данными ссылочного типа, то по умолчанию в каждой ячейке записаны null

```
1  String[] seasons  = new String[4]; /* объявили и создали массив. Java выделила память под массив из 4
2
3  seasons[0] = "Winter"; /* в первую ячейку, то есть, в ячейку с нулевым номером мы записали строку Wint
4  seasons[1] = "Spring"; // проделываем ту же процедуру с ячейкой номер 1 (второй)
5  seasons[2] = "Summer"; // ...номер 2
6  seasons[3] = "Autumn"; // и с последней, номер 3
```

Теперь во всех четырёх ячейках нашего массива записаны названия сезонов. Инициализацию также можно провести по-другому, совместив с инициализацией и объявлением:

```
1  String[] seasons  = new String[] {"Winter", "Spring", "Summer", "Autumn"};
```

Более того, оператор `new` можно опустить:

```
1  String[] seasons  = {"Winter", "Spring", "Summer", "Autumn"};
```

Массивы

Тип данных [] arr = new Тип данных [размерность]

int[] a = new int[50]

или

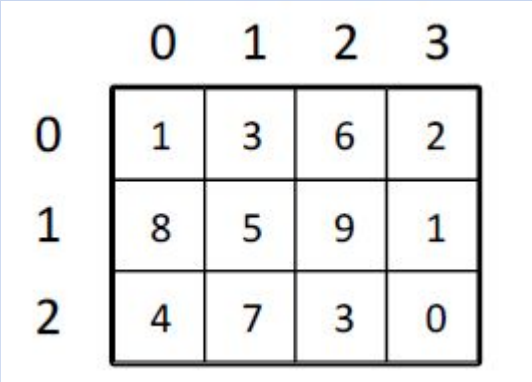
Int[ ] arr = { 1,2,3 }

```
int[,] twoDimensionalArray;
```

```
int[,,] threeDimensionalArray;
```

Двумерные массивы

```
int[,] intMatrix = new int[3, 4];
```



	0	1	2	3
0	1	3	6	2
1	8	5	9	1
2	4	7	3	0

Let's examine the next example:

```
int[,] matrix =
{
   {1, 2, 3, 4},
   {5, 6, 7, 8},
};
```

The array **matrix** has 8 elements, stored in 2 rows and 4 colum
element can be accessed in the following way:

```
matrix[0, 0]  matrix[0, 1]  matrix[0, 2]  matrix[0, 3]
matrix[1, 0]  matrix[1, 1]  matrix[1, 2]  matrix[1, 3]
```

In this example we can access each element using indices. If we a
index for rows to **row**, and the index for columns to **col**, then we ca
any element as shown:

```
matrix[row, col]
```

Двумерные массивы

Printing Matrices – Example

In the next example we will demonstrate how we can print two-d arrays to the console:

```
// Declare and initialize a matrix of size 2 x 4
int[,] matrix =
{
   {1, 2, 3, 4}, // row 0 values
   {5, 6, 7, 8}, // row 1 value
};

// Print the matrix on the console
for (int row = 0; row < matrix.GetLength(0); row++)
{
   for (int col = 0; col < matrix.GetLength(1); col++)
   {
      Console.Write(matrix[row, col]);
   }
   Console.WriteLine();
}
```

Задачи на массивы

1. Напишите программу, которая создает массив из 20 элементов типа integer и инициализирует каждый из элементов значением, равным индексу элемента, умноженному на 5. Распечатайте элементы в консоли.
2. Напишите программу, которая считывает с консоли два массива и проверяет, равны ли они (два массива равны, если они имеют одинаковую длину и все их элементы, имеющие одинаковый индекс, равны).
3. Напишите программу, которая сравнивает два массива типа char лексикографически (посимвольно) и проверяет, какой из них является первым в лексикографическом порядке.

“CAR”>”CAA”

Задачи на массивы

Напишите программу, которая находит максимальную последовательность последовательно расположенных возрастающих целых чисел. Пример: {3, 2, 3, 4, 2, 2, 4} □ {2, 3, 4}.

Напишите программу, которая находит максимальную последовательность возрастающих элементов в массиве arr [n]. Нет необходимости размещать элементы последовательно. Например: {9, 6, 2, 7, 4, 7, 6, 5, 8, 4} □□ {2, 4, 6, 8}.

Напишите программу, которая находит наиболее часто встречающийся элемент в массиве. Пример: {4, 1, 1, 4, 2, 3, 4, 4, 1, 2, 4, 9, 3} □ 4 (5 раз).
Напишите программу для поиска последовательности номеров соседей в массиве, который имеет сумму определенного числа S. Пример: {4, 3, 1, 4, 2, 5, 8}, S = 11 □ {4, 2, 5}.

Циклы по коллекции

Iteration with "foreach" Loop

One of the most used constructs for iterating through elements of an array is **foreach**. The **foreach-loop construct in C#** is as follows:

```
foreach (var item in collection)
{
   // Process the value here
}
```

Iteration with "foreach" Loop – Example

In the next example we will learn how to use the **foreach** loop to **iterate through the array**:

```
string[] capitals =
   { "Sofia", "Washington", "London", "Paris" };

foreach (string capital in capitals)
{
   Console.WriteLine(capital);
}
```

Классы

- *Классы* позволяют проводить конструирование из полезных компонентов, обладающих простыми инструментами, что позволяет абстрагироваться от деталей реализации.
- Данные и операции над ними образуют определенную сущность, и они не разносятся по всей программе, как нередко бывает в случае процедурного программирования, а описываются вместе. Локализация кода и данных улучшает наглядность и удобство сопровождения программного обеспечения.
- *Инкапсуляция* позволяет привнести свойство *модульности*, что облегчает распараллеливание выполнения задачи между несколькими исполнителями и обновление версий отдельных компонентов.

Полиморфизм оказывается полезным преимущественно в следующих ситуациях.

- Обработка разнородных структур данных. Программы могут работать, не различая вида *объектов*, что существенно упрощает код. Новые виды могут быть добавлены в любой момент.
- Изменение *поведения* во время исполнения. На этапе исполнения один *объект* может быть заменен другим, что позволяет легко, без изменения кода, адаптировать алгоритм в зависимости от того, какой используется *объект*.
- Реализация работы с наследниками. Алгоритмы можно обобщить настолько, что они уже смогут работать более чем с одним видом *объектов*.
- Создание "каркаса" (framework). Независимые от приложения части предметной области могут быть реализованы в виде набора универсальных *классов*, или каркаса (framework), и в дальнейшем расширены за счет добавления частей, специфичных для конкретного приложения.