

C# II Arrays

Plan

- What is it?

1. Одномерные

2. Двумерные (3-х, 4-х,...)

3. Рваные

4. Действия с массивами



What is IT?

- Много однотипных данных
- Хотим хранить, изменять и передавать вместе
- Reference Type

```
int[] arr = null;
```

C#

||

Stack & Heap

Stack

- Особая область памяти, в которой хранятся локальные переменные и адреса возврата из методов.
- Область в оперативной памяти, выделяемая операционной системой для потока приложения.
- У приложения может быть запущено несколько потоков в одном процессе.
- Размер Stack'a порядка 1 Mb

Heap

- Вся остальная область памяти, для хранения массивов и других объектов.



Спасибо за внимание!

1Dimensional

Отсчет ведется с
нуля!

Объявление и инициализация

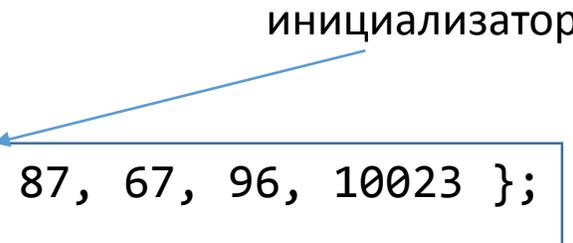
```
int[] arr; // объявление переменной, содержащей массив
```

```
arr = new int[100]; // инициализация (выделение памяти в куче для ста элементов)
```

0	1	2	3	4	5	6	7	8	9	...	99
45	334	87	-7	87	0	65	87	-56	678		90

Упрощенный синтаксис

```
int[] arr = new int[9] { 10, 31, 62, 45, 45, 87, 67, 96, 10023 };
```



```
int[] arr = new int[] { 10, 31, 62, 45, 45, 87, 67, 96, 10023 };
```

```
int[] arr = { 10, 31, 62, 45, 45, 87, 67, 96, 10023 };
```

Обход всех элементов массива

```
for(int i = 0; i < arr.Length; i++)    foreach(var item in arr)
{                                       {
    Console.WriteLine(arr[i]);         Console.WriteLine(item);
}
```

`arr.Length` // длина массива (количество элементов в массиве)

`foreach` // только на чтение (нельзя изменить элемент массива)

Копирование массива

```
// т.к. массивы – ссылочный тип, иногда возникает необходимость в их копировании
// Функция CopyTo копирует arr в array начиная с элемента с индексом index
arr.CopyTo(Array array, int index);
```

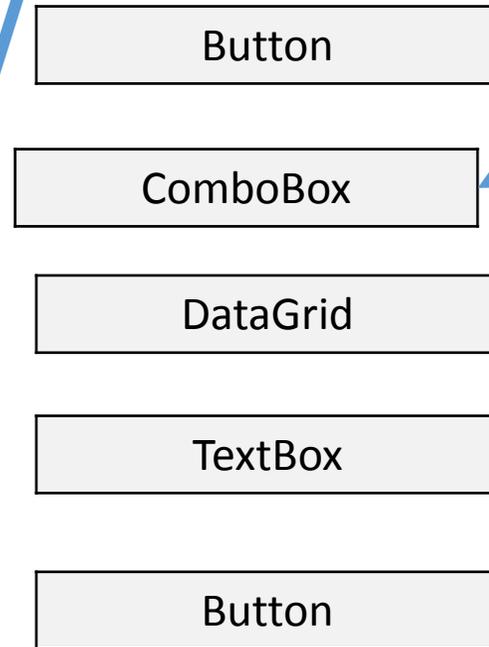
Массивы в куче

Отсчет ведется с
нуля!

99	Int32
98	Int32
97	Int32
96	Int32
⋮	⋮
3	Int32
2	Int32
1	Int32
0	Int32
	Дополнительная информация

Значимый тип

```
Int32[] myIntegers;  
myIntegers = new Int32[100];
```



Ссылочный тип

```
Control[] myControls;  
myControls = new Control[60];
```

Элемент управления	59
Элемент управления	58
Элемент управления	57
Элемент управления	56
⋮	⋮
Элемент управления	3
Элемент управления	2
Элемент управления	1
Элемент управления	0
Дополнительная информация	

2D

Объявление и инициализация

```
int[,] arr2D = new int[num_rows, num_cols];
```

```
arr2D.GetLength(int dimension); // количество элементов в заданном измерении
```

```
arr2D.Rank; // размерность массива
```

Обход всех элементов массива

```
for (int i = 0; i < arr2D.GetLength(0); i++)  
{  
    for (int j = 0; j < arr2D.GetLength(1); j++)  
    {  
        Console.WriteLine(arr2D[i, j]);  
    }  
}
```

	0	1	2	3	4
0	45	-67	78	-76	87
1	76	56	8	67	67
2	56	56	76	-45	45
3	45	676	76	456	56
4	345	65	45	566	68
5	45	65	67	97	87

ND

```
int[, ,] arr3D = new int[d0_length, d1_length, d2_length];
```

Можно задать, чтобы отсчет велся не с нуля (для 2D тоже)

```
arr.GetLowerBound(int dimension); // нижняя граница измерения  
arr.GetUpperBound(int dimension); // верхняя граница измерения
```

```
for (int x = arr3D.GetLowerBound(0); x < arr3D.GetUpperBound(0); x++)  
{  
    for (int y = arr3D.GetLowerBound(1); y < arr3D.GetUpperBound(1); y++)  
    {  
        for (int z = arr3D.GetLowerBound(2); z < arr3D.GetUpperBound(2); z++)  
        {  
            Console.WriteLine(arr3D[x,y,z]);  
        }  
    }  
}
```

[https://msdn.microsoft.com/ru-ru/library/x836773a\(v=vs.110\).aspx](https://msdn.microsoft.com/ru-ru/library/x836773a(v=vs.110).aspx)

Ragged (рваные) Arrays

Массивы в массиве

**Длины вложенных массивов
могут быть различны**

```
int[][] rvan = new int[4][];
```

```
rvan[0] = new int[5];
```

```
rvan[1] = new int[12];
```

```
rvan[2] = new int[120];
```

```
rvan[3] = new int[0];
```

Rectangular Array

[0,0]	[0,1]	[0,2]	[0,3]
[1,0]	[1,1]	[1,2]	[1,3]
[2,0]	[2,1]	[2,2]	[2,3]

Ragged Array

[0] [0]	[0] [1]	[0] [2]	
[1] [0]	[1] [1]		
[2] [0]	[2] [1]	[2] [2]	[2] [3]

Передача в функцию

Поскольку передается ссылка, функция может изменять элементы массива

```
void Main()
{
    int[] arr = new int[5] { 1, 2, 3, 4, 5 };
    ChangeFirstLast(arr);
    Console.WriteLine(arr);
}
```

```
void ChangeFirstLast(int[] arr)
{
    int first = arr[0];
    arr[0] = arr[arr.Length - 1];
    arr[arr.Length - 1] = first;
}
```

В чем уязвимость функции?

Если функция возвращает массив, то рекомендуется всегда возвращать инициализированный массив даже если он не содержит элементов (вместо null)

Присваивание

```
string[] womans = new string[5] { "Sasha", "Masha", "Dasha", "Glasha", "Vika" };
```

```
string[] mans = new string[6] { "Egor", "Andrew", "Richard", "Sergei", "Yurii", "Anton" };
```

```
womans = mans;  
mans[0] = "Viktor";
```

```
Console.WriteLine(mans[0]); // ? Viktor
```

```
Console.WriteLine(womans[0]); // ? Viktor
```

```
womans[5] = "Yulia";
```

```
Console.WriteLine(mans[5]); // ? Yulia
```

```
Console.WriteLine(womans[5]); // ? Yulia
```

womans и *womans* ссылаются на один и тот же массив [mans](#)

Передача значимых типов по ссылке

Ключевые слова `ref` и `out` используются для передачи аргументов по ссылке.

- `ref` требует инициализации переменной перед ее передачей,
- `out` требует, чтобы переменной внутри функции обязательно присвоилось значение.

```
static double FindMax(double[] arr, out int index)
{
    double max = arr[0];
    index = 0;
    for (int i = 0; i < arr.Length; i++)
    {
        if (arr[i] > max)
        {
            index = i;
            max = arr[i];
        }
    }
    return max;
}

int i;
FindMax(new double[] { 3, 5.6, 75.99, 65, 6 }, out i);
```



Спасибо за внимание!