Массивы

- ***Массивом*** называют упорядоченную совокупность элементов одного типа.
- Каждый элемент *массива* имеет один или несколько индексов, определяющих порядок элементов.
- Количество индексов характеризует ***размерность массива***. Каждый индекс изменяется в диапазоне от нуля до некоторого числа N>0. Индексы задаются только целочисленным типом.

- *Массивы* относятся к ссылочным типам, а следовательно, память им отводится в "куче".
- В языке C# имеются одномерные массивы и многомерные массивы.
- Кроме них, в языке C# также имеется новый тип массивов – ступенчатый.

Одномерные массивы

Массивы соответствуют типу System.Array

Объявление массива *<type>* **[ ]** *<имя_массива>* ;

<type> тип элемента массива
[] признак того, что объявляется массив
<имя_массива> идентификатор массива

Пример

`int k;` переменная **k**, представляющая целое число,
значение не задано, переменная создаётся в стеке.

`int[] m;` массив **m**, состоящий из целых чисел,
число и значения элементов не заданы,
в стеке создаётся пустая ссылка.

`double f;` переменная **f**, представляющая вещественное число,
значение не задано, переменная создаётся в стеке.

`double[] x;` массив **X**, состоящий из вещественных чисел,
число и значения элементов не заданы,
в стеке создаётся пустая ссылка.

Создание массива *имя_массива = new type[n];*

имя_массива = new type[n] {x_1, x_2, …, x_n};

Если уже выполнено

```
int[] m;
double[] x;
```

то можно создать массив в куче:

`m=new int[5];` создаётся в куче набор из 5 **пустых** элементов целого типа `m[0], m[1], m[2], m[3], m[4]`

`x=new double[10];` создаётся в куче набор из 10 **пустых** элементов. веществ. типа `x[0], x[1], x[2], ..., x[9]`

```
m=new int[5] {1, 2, 0, 5, -7};
```

создаётся в куче набор из 5 элементов целого типа равных указанным числам `m[0]=1, ..., m[4]=-7`

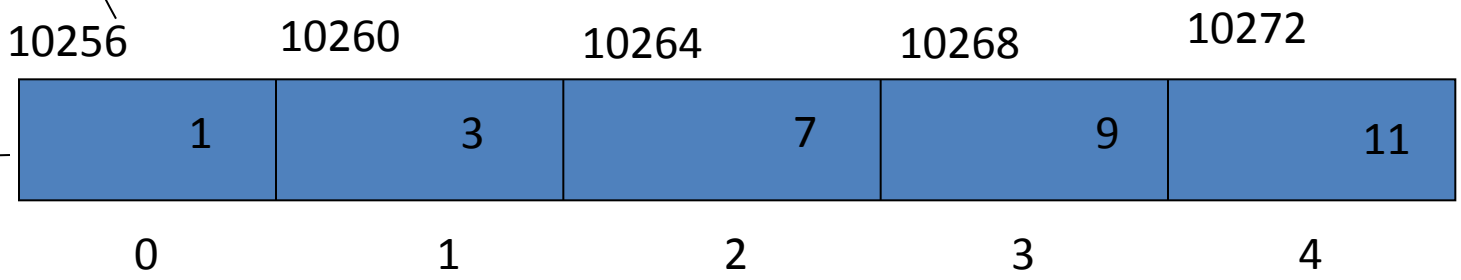
Массив - Array

int [ ] a = new int [5]; // можно одним оператором

a[ ] =0; // нельзя использовать массив без индекса

a[3] = 9;

a[0] = 1



Инициализация массивов

- По умолчанию записывается нуль или null.
- Для работы с массивом необходимо заполнить его значения:

```
int[] a = new int [5] {1,2,3,4,5};
int[] b = {5,7,9};

char[] s = new char [7] {'s','t','u','d','e','n','t'};
char[] s = {'s','t','u','d','e','n','t'};
```

Пример разных способов объявления *массивов*

```
//объявляется одномерный массив A
int[] A = new int[5] {1,2,3,4,5};
//объявление массива x с явной
   инициализацией
int[] x ={5,5,6,6,7,7};
//объявление массивов с отложенной
   инициализацией
int[] u,v;
u = new int[3];
```

Пример работы с массивами

```
int[] u,v;

u = new int[3];

for (int i=0; i<3; i++) u[i] = i+1;
v = new int[4];
v=u; //допустимое присваивание
```

- Оператора присваивания `v = u` является правильным ссылочным присваиванием: хотя `u` и `v` имеют разное число элементов, но они являются объектами одного класса.
- Теперь обе ссылки `u` и `v` будут теперь указывать на один и тот же *массив*, так что изменение элемента одного *массива* немедленно отразится на другом *массиве*.
- На *массив* `v` теперь никто ссылаться не будет, и он будет считаться мусором, который автоматически удаляется с помощью сборщика мусора.

Методы и свойства массивов

Свойства массива

- Rank – количество размерностей
- Length – количество элементов массива (32 битное значение)

Статические методы класса System.Array

- BinarySearch() – индекс заданного элемента в сортированном массиве
- IndexOf() – индекс заданного элемента в массиве
- Clear() – обнуление группы элементов в массиве
- Clone( ) – неполное копирование (если элементы массива ссылки, то копируются ссылки)
- Copy() – копирует все (или часть) элементов одного массива в другой
- Resize() – изменение длины массива
- Reverse() – изменение порядка всех (или части) элементов на обратный
- Sort() – сортировка элементов массива по возрастанию

Методы массива

- CopyTo( ) – копирует часть элементов одного массива в другой
- GetLength (n) – количество элементов в размерности n

Пример

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

Оператор	M[0]	M[1]	M[2]	M[3]	M[4]	M[5]
`int[] m = new int[5] { 1, 2, 3, 4, 5 };`	1	2	3	4	5	
`Console.WriteLine(m.Length);`	5					
`Console.WriteLine(m.Rank);`	1					
`Console.WriteLine(Array.BinarySearch(m, 4));`	3					
`Console.WriteLine(Array.BinarySearch(m, 7));`	-6					
`Console.WriteLine(Array.IndexOf(m, 4));`	3					
`Console.WriteLine(Array.IndexOf(m, 7));`	-1					
`Array.Clear(m, 2, 2);`	1	2	0	0	5	
`m[2] = -8;`	1	2	-8	0	5	
`int[] mm = (int[])m.Clone();`	1	2	-8	0	5	
`Array.Resize(ref m, 6);`	1	2	-8	0	5	0
`Array.Reverse(m);`	0	5	0	-8	2	0
`Array.Sort(m);`	-8	0	0	0	2	5

Использование цикла **foreach** с массивами

Традиционный способ обработки массивов:

Найти сумму элементов массива

```
int[] ar =  { 1, 2, 3, 4, 5 };
int s=0;
for (int i = 0; i<ar.Length; i++)  s += ar[i];
```

Оператор foreach

foreach(<тип> <переменная> in <имя массива>) оператор

- <тип> переменной должен соответствовать типу элементов массива.
- на каждом шаге переменная цикла получает значение очередного элемента в соответствии с порядком, установленным на элементах массива.
- с этим текущим элементом и выполняется тело цикла - выполняется столько раз, сколько элементов находится в массиве.
- цикл заканчивается, когда полностью перебраны все элементы массива.

Пример использования foreach

Найти сумму элементов массива

```
int[] arr = new int[5] {1,2,3,4,5}
int s =0;
foreach(int c in arr)
{
    s += c;
}
```

Примеры работы с одномерными массивами

Выполнить циклический сдвиг элементов влево на один

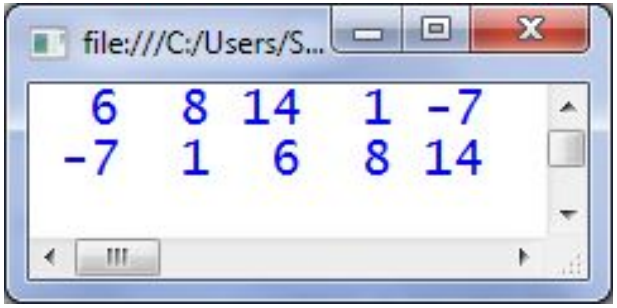
```
int[] m = new int[5] { 1, 2, 3, 4, 5 };

int x = m[0];
for (int i = 0; i < m.Length-1; i++)
      m[i]=m[i+1];
m[m.Length - 1] = x;
```

Выполнить сортировку элементов массива.

(простейший алгоритм)

```
// перебираем все пары элементов
for (int i = 0; i < m.Length-1; i++)        //  левый в паре
    for (int j = i + 1; j < m.Length; j++) // правый в паре
        if (m[i] > m[j])                    // левый больше правого
        {
            int x = m[i];
            m[i] = m[j];
            m[j] = x;
        }
```



Заменить каждый элемент массива суммой данного и предшествующих элементов

1 2 3 4 5 -> 1 3 6 10 15

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

```
for (int i = 0; i < m.Length; i++) {
    int s=0;
    for (int j = 0; j <= i; j++) s+=m[j];
    m[i]=s;
}
```

1 2 3 4 5 -> 1 3 7 15 31

```
for (int i = m.Length-1; i >= 0; i--)
{
    int s=0;
    for (int
    m[i]=s;
}
```

```
for (int i = 1; i < m.Length; i++)
      m[i] += m[i-1];
```

1 2 3 4 5 -> 1 3 6 10 15

Многомерные массивы. Двухмерные массивы (таблицы)

- Многомерные массивы характеризуются двумя или более измерениями, а доступ к каждому элементу обеспечивается двумя или более индексами.
- Простейшим многомерным массивом является двумерный массив.
- Двумерные массивы создаются с помощью инструкций следующего типа:

тип [,] имя_массива = **new** *тип [размер1,размер2];*

Пример

```
int [,] a = new int [3,4];
a [1,2] = 4;

int [,] a = {{1,0,1,0}, {2,3,4,5}, {0,3,0,3}};
```

	0	1	2	3
0	1	0	1	0
1	2	3	4	5
2	0	3	0	3

10256

1	0	1	0	2	3	4	5	0	3	0	3

Пример создания двумерного массива

- Пример объявления и инициализации двумерного массива:

  int [,] ms = new int [2,3] {{1,2,3},{4,5,6}};
- Доступ к элементам массива :

  int k = ms[1,2]; // получим значение 6
- Пример: найти сумму всех элементов двухмерного массива:

```
int s = 0;
for (int i = 0; i < 2; i++)
  for (int j = 0; j < 3; j++)
    s += ms[i, j];
```

- В результате выполнения этого кода s получит значение 21 – сумму всех элементов массива ms.

Выполнить умножение матрицы размером 3 x 3 на столбец из трёх элементов

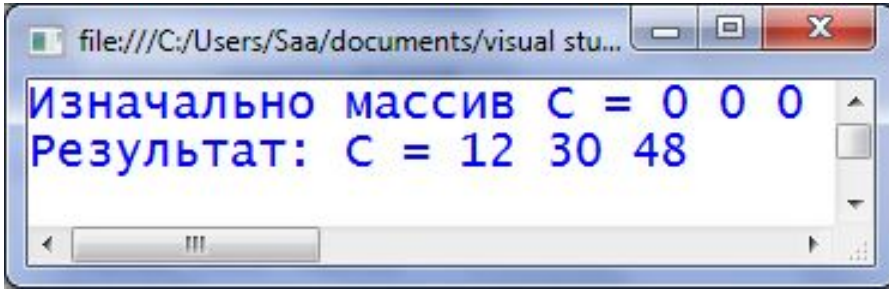
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

$$c_i = a_{i1}b_1 + a_{i2}b_2 + a_{i3}b_3 = \sum_{j=1}^{3} a_{ij}b_j$$

Выполнить умножение матрицы размером 3 x 3 на столбец из трёх элементов

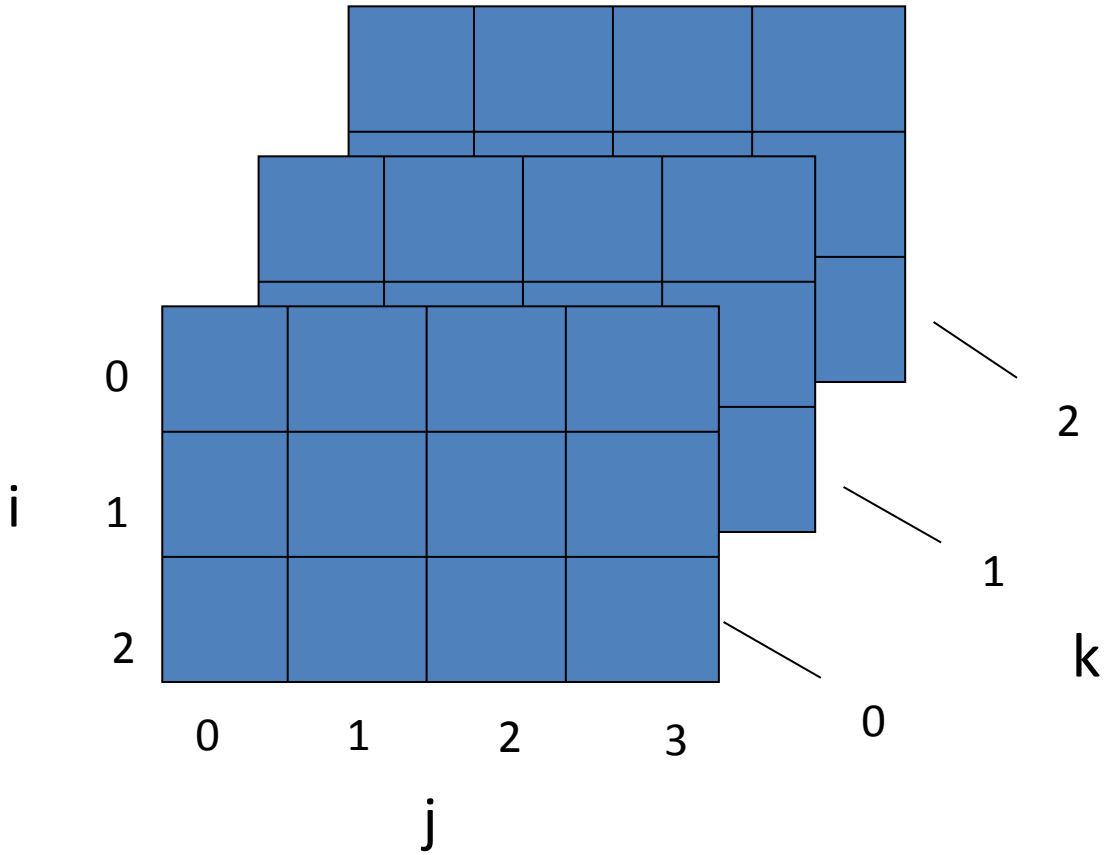
$$c_i = \sum_{j=1}^{3} a_{ij} b_j$$

```
//объявление и инициализация двумерного массива A
int[,] a = {  {1,2,3},  {4,5,6},  {7,8,9} };
//объявление и инициализация одномерного массива B
int[] b = { 2, 2, 2 };
//объявление одномерного массива C; по умолчанию инициализируется нулями
int[] c = new int[3];
//вывод массива C
Console.WriteLine("Изначально массив C = {0} {1} {2}", c[0], c[1], c[2]);
//программа умножения матрицы на столбец
for (int i = 0; i < 3; i++)
{
    c[i] = 0;
    for (int j = 0; j < 3; j++)
        c[i] = c[i] + a[i, j] * b[j];
}
//вывод результатов
Console.WriteLine("Результат: C = {0} {1} {2}", c[0], c[1], c[2]);
```



Многомерные массивы - трехмерные

```
float [,,] m = new float [3,4,3];
m [i,j,k] = 3;
```



Ступенчатые массивы

Массив массивов – количество элементов по каждому измерению непостоянно.

Например:

```
int[][] d = new int[3][];
```

Для создания самих строк массива нужно создать объекты соответствующих типов:

```
d[0] = new int[2] { 1, 2 };
d[1] = new int[6] { 3, 4, 5, 6 };
d[2] = new int[3] { 7, 8, 9 };
```

Схема данного ступенчатого массива: