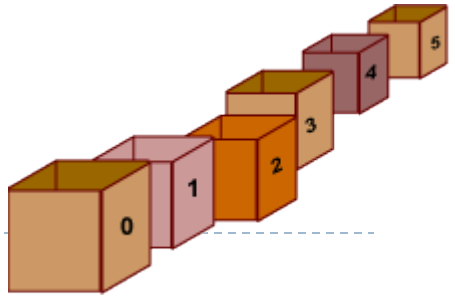
Программирование C#

4. Массивы и строки

Карбаев Д.С., 2016

Одномерные массивы



- Массив представляет собой совокупность переменных одного типа с общим для обращения к ним именем.
- Одномерный массив представляет собой список связанных переменных.
- общая форма:

```
тип[] имя_массива = new тип[размер];
```

где тип объявляет конкретный тип элемента массива. Тип элемента определяет тип данных каждого элемента, составляющего массив.

- Пример:

```
int[] sample = new int[10];
```

- В переменной sample хранится ссылка на область памяти, выделяемой для массива оператором new. Эта область памяти должна быть достаточно большой, чтобы в ней могли храниться десять элементов массива типа int. Приведенное выше объявление массива можно разделить на два отдельных оператора. Например:

```
int[] sample;
sample = new int[10];
```

Одномерный массив

```
static void Main()        {
    int[] sample = new int[10];
    int i;
    for (i = 0; i < 10; i = i + 1)
        sample[i] = i;
    for (i = 0; i < 10; i = i + 1)
        Console.WriteLine("sample[" + i + "]: "
                          + sample[i]);
}
```

- Результат

```
sample[0]: 0
sample[1]: 1
sample[2]: 2
sample[3]: 3
sample[4]: 4
sample[5]: 5
sample[6]: 6
sample[7]: 7
sample[8]: 8
sample[9]: 9
```

Среднее арифметическое ряда значений

```
        static void Main() {
            int[] nums = new int[10];
            int avg = 0;
            nums[0] = 99;
            nums[1] = 10;
            nums[2] = 100;
            nums[3] = 18;
            nums[4] = 78;
            nums[5] = 23;
            nums[6] = 63;
            nums[7] = 9;
            nums[8] = 87;
            nums[9] = 49;
            for (int i = 0; i < 10; i++)
                avg += nums[i];
            avg = avg / 10;
            Console.WriteLine("Среднее: " + avg);
        }
```

- Результат

```
Среднее: 53
```

Инициализация массива

- Общая форма инициализации одномерного массива:

```
тип[] имя_массива = {val1, val2, val3, ..., valN};
```

- где val1-valN обозначают первоначальные значения, которые присваиваются по очереди, слева направо и по порядку индексирования.
- Для хранения инициализаторов массива в C# автоматически распределяется достаточный объем памяти. А необходимость пользоваться оператором new явным образом отпадает сама собой.
- Пример:

```
int[] nums = {99, 10, 100, 18, 78, 23, 63, 9, 87, 49};
```

Инициализация массива

- Приведенный ниже фрагмент кода считается верным, но избыточным для инициализации массива nums в упомянутой выше программе.

```
int[] nums = new int[] {99, 10, 100, 18, 78, 23, 63, 9, 87, 49};
```

- New можно использовать, если новый массив присваивается уже существующей переменной ссылки на массив:

```
int[] nums;
nums = new int[] {99, 10, 100, 18, 78, 23, 63, 9, 87, 49};
```

- При инициализации массива его размер можно указывать явным образом, но этот размер должен совпадать с числом инициализаторов

```
int[] nums = new int[10] {99, 10, 100, 18, 78, 23, 63, 9, 87,
49};
```

Соблюдение границ массива

```
static void Main() {
    int[] sample = new int[10];
    int i;
    // Воссоздать превышение границ массива.
    for (i = 0; i < 100; i = i + 1)
        sample[i] = i;
}
```

Как только значение переменной i достигнет 10, возникнет исключительная ситуация типа IndexOutOfRangeException, связанная с выходом за пределы индексирования массива, и программа преждевременно завершится

Двумерный массив

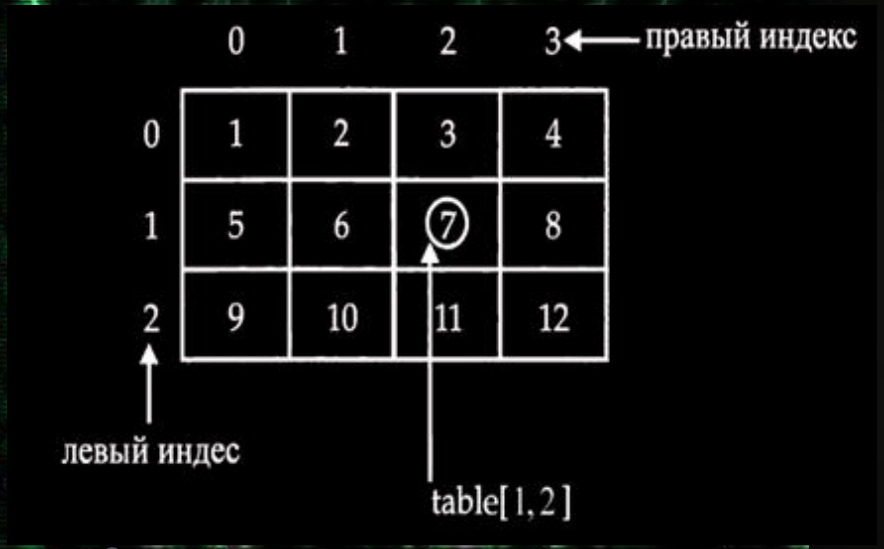
- В следующей строке кода объявляется двумерный массив целых чисел размерами 10x20.

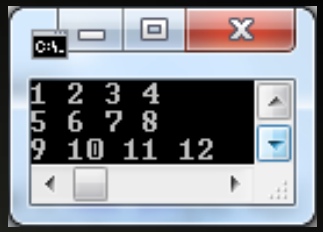
```
int[,] table = new int[10, 20];
```

- Для доступа к элементу двумерного массива следует указать оба индекса, разделив их запятой

```
table[3, 5] = 10;
```

- Пример:






```
static void Main()
{
    int t, i;
    int[,] table = new int[3, 4];
    for (t = 0; t < 3; ++t)
    {
        for (i = 0; i < 4; ++i)
        {
            table[t, i] = (t * 4) + i + 1;
            Console.Write(table[t, i] + " ");
        }
        Console.WriteLine();
    }
}
```

Многомерные массивы

- Общая форма объявления многомерного массива.

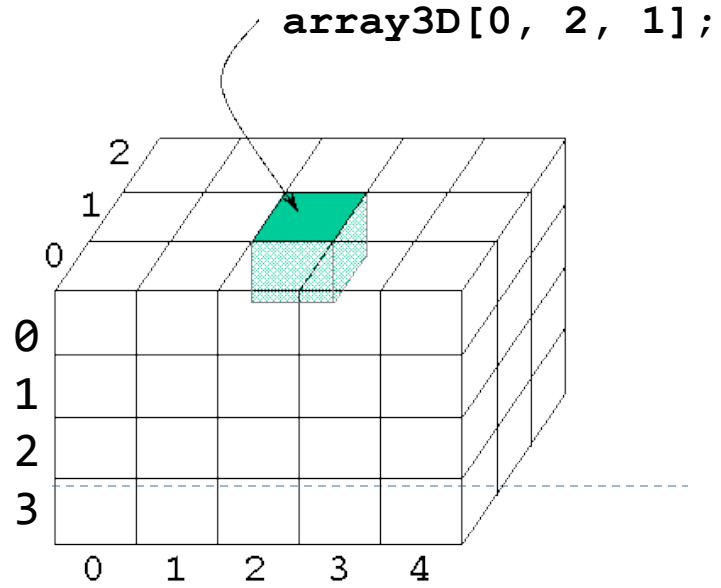
```
тип[, ..., ] имя_массива = new тип[размер1, размер2, ... размеры];
```

- Например, в приведенном ниже объявлении создается трехмерный целочисленный массив размерами 4x5x3.

```
int[,,] array3D = new int[4, 5, 3];
```

- В следующем операторе элементу массива array3D с координатами местоположения 0,2,1) присваивается значение 100.

```
array3D[0, 2, 1] = 100;
```



Суммировать значения по одной из диагоналей

```
static void Main() {
    int[, ,] m = new int[3, 3, 3];
    int sum = 0;
    int n = 1;
    for (int x = 0; x < 3; x++)
        for (int y = 0; y < 3; y++)
            for (int z = 0; z < 3; z++)
                m[x, y, z] = n++; //заполенение матрицы
   sum = m[0, 0, 0] + m[1, 1, 1] + m[2, 2, 2]; //сумммирование
    Console.WriteLine("Сумма значений по первой диагонали: " + sum);
}
```

Результат:

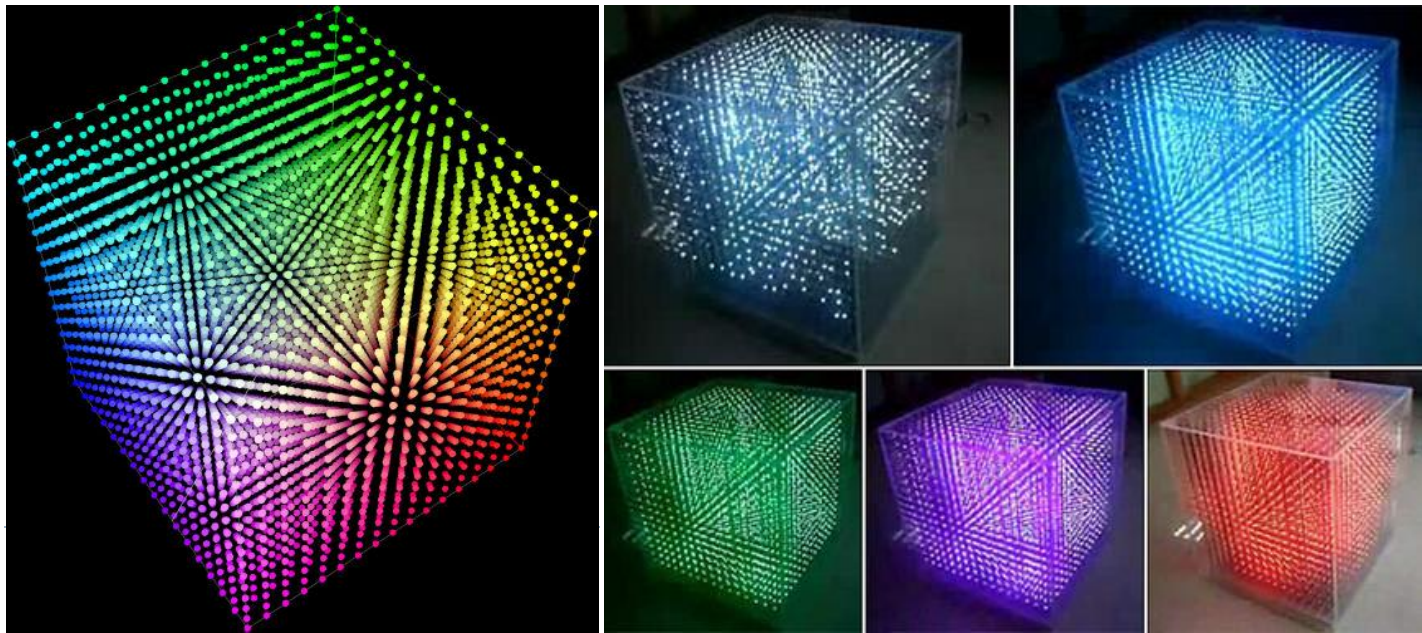
```
Сумма значений по первой диагонали: 42
```

Инициализация многомерных массивов

- Общая форма инициализации двумерного массива:

```
тип[,] имя_массива = {
{val, val, val, ..., val},
{val, val, val, ..., val},
{val, val, val, ..., val}
};
```

где val обозначает инициализирующее значение, а каждый внутренний блок — отдельный ряд.



Инициализация двумерного массива

```
static void Main() {
      int[,] sqrs = {
      { 1, 1 },
      { 2, 4},
      { 3, 9 },
      { 4, 16 },
      { 5, 25 },
      { 6, 36 },
      { 7, 49 },
      { 8, 64 },
      { 9, 81 },
      { 10, 100 }
      };
      int i, j;
      for (i = 0; i < 10; i++) {//вывод массива
          for (j = 0; j < 2; j++)
              Console.Write(sqrs[i, j] + " ");
          Console.WriteLine();
      }
  }
```

результат:

```
1 1
2 4
3 9
4 16
5 25
6 36
7 49
8 64
9 81
10 100
```

Ступенчатые массивы

Two-Dimensional Array

1	2	3
4	5	6
7	8	9

Jagged Array

1	2				
3	4	5	6	7	8
9	10	11			

- Для объявления двумерного ступенчатого массива служит следующая общая форма:

```
тип[] [] имя_массива = new тип [размер] [];
```

где размер обозначает число строк в массиве.

```
int[][] jagged = new int[3][];
jagged[0] = new int[4];
jagged[1] = new int[3];
jagged[2] = new int[5];
```

- После выполнения этого фрагмента кода массив jagged выглядит так, как показано ниже.

```
jagged [0][0]  jagged [0][1]  jagged [0][2]  jagged [0][3]
jagged [1][0]  jagged [1][1]  jagged [1][2]
jagged [2][0]  jagged [2][1]  jagged [2][2]  jagged [2][3]  jagged [2][4]
```

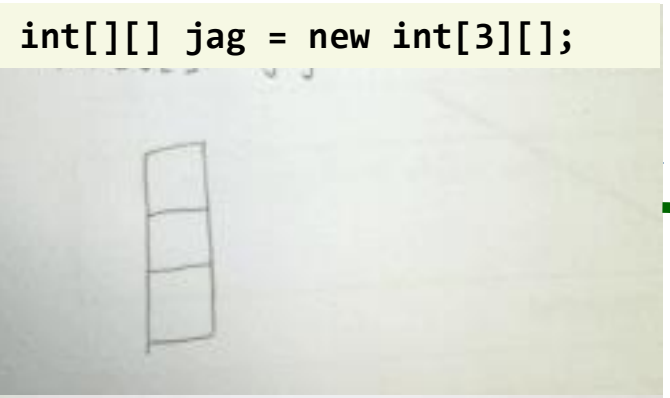
- Присвоение значения:

```
jagged[2][1] = 10;
```

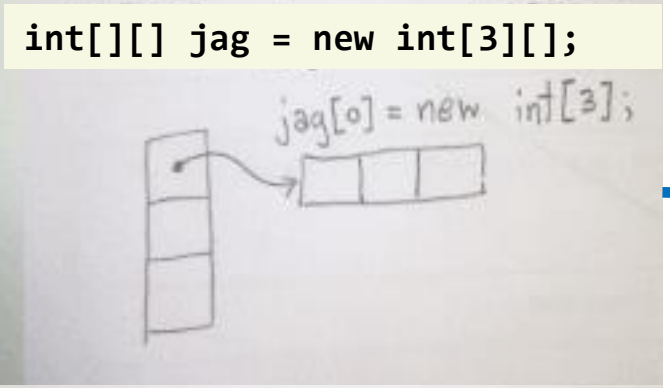
Ступенчатый массив

Двумерный массив

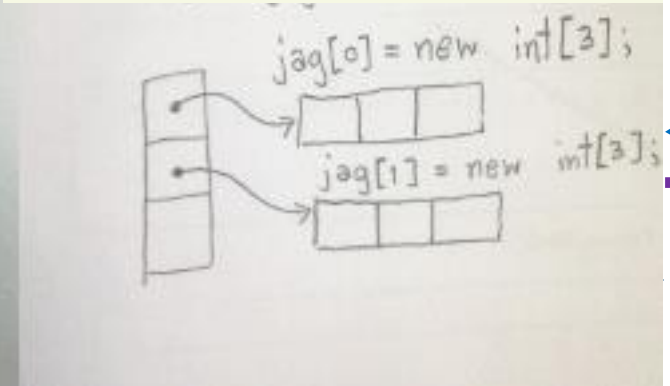
```
int[][] jag = new int[3][];
```

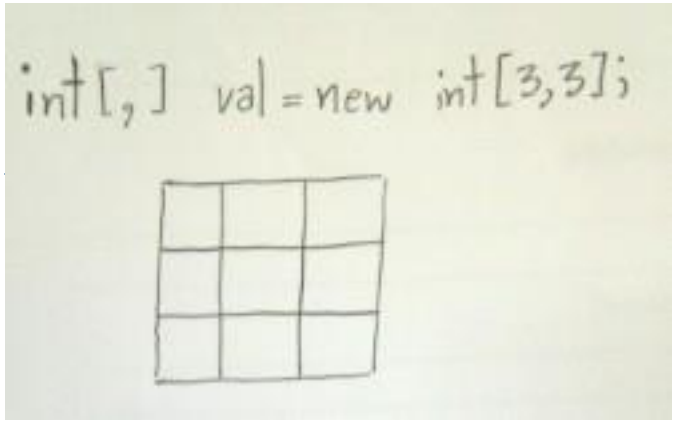


```
int[][] jag = new int[3][];
```

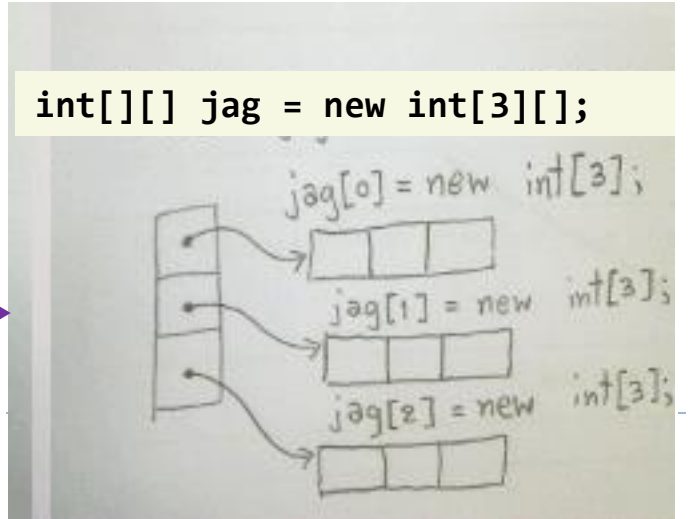



```
int[][] jag = new int[3][];
```






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



Ступенчатые массивы

```
static void Main() {
    int[][] jagged = new int[3][]; int i;
    jagged[0] = new int[4];jagged[1] = new int[3];
    jagged[2] = new int[5];
    // Сохранить значения в нулевом массиве.
    for (i = 0; i < 4; i++) jagged[0][i] = i;
    // Сохранить значения во первом массиве.
    for (i = 0; i < 3; i++) jagged[1][i] = i;
    // Сохранить значения в втором массиве.
    for (i = 0; i < 5; i++) jagged[2][i] = i;
    // Вывести значения из нулевого массива.
    for (i = 0; i < 4; i++)
        Console.Write(jagged[0][i] + " ");
    Console.WriteLine();
    // Вывести значения из первого массива.
    for (i = 0; i < 3; i++)
        Console.Write(jagged[1][i] + " ");
    Console.WriteLine();
    // Вывести значения из второго массива.
    for (i = 0; i < 5; i++)
        Console.Write(jagged[2][i] + " ");
    Console.WriteLine();  }
```

Результат.

```
0 1 2 3
0 1 2
0 1 2 3 4
```

Ступенчатые массивы представляют собой массивы массивов, и поэтому они не обязательно должны состоять из одномерных массивов.
Создается массив двумерных массивов:

```
int[] [,] jagged = new int[3] [,];
```

Элементу массива jagged [0] присваивается ссылка на массив размерами 4x2:

```
jagged[0] = new int [4, 2];
```

Элементу массива jagged [0] [1,0] присваивается значение переменной i.

```
jagged[0] [1,0] = i;
```

Присваивание ссылок на массивы

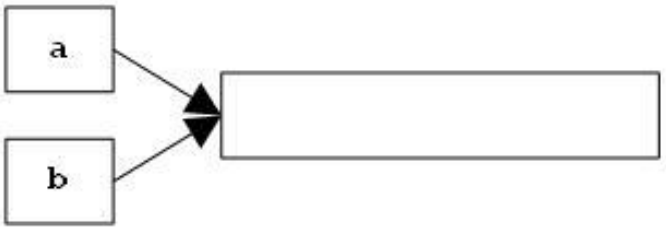
Объявить сслыки

Создать массивы

```
int [] a;
bool[] b;

a = new int [5];
b = new bool[3];
```

Присвоить ссылку на массив

```
int[] a = new int[5];
int[] b;

b = a;
...
```



Присваивание ссылок на массивы

```
static void Main() {
    int i;
    int[] nums1 = new int[10];
    int[] nums2 = new int[10];
    for (i = 0; i < 10; i++) nums1[i] = i;
    for (i = 0; i < 10; i++) nums2[i] = -i;
    Console.Write("Содержимое массива nums1: \n");
    for (i = 0; i < 10; i++)
        Console.Write(nums1[i] + " ");
    Console.WriteLine();
    Console.Write("Содержимое массива nums2: ");
    for (i = 0; i < 10; i++)
        Console.Write(nums2[i] + " ");
    Console.WriteLine();

  // см. далее…
```

Результат:

```
Содержимое массива nums1:
0 1 2 3 4 5 6 7 8 9

Содержимое массива nums2:
0 -1 -2 -3 -4 -5 -6 -7 -8 -9
```

Присваивание ссылок на массивы

```
// …продолжение
 // теперь nums2 ссылается на nums1
 nums2 = nums1;
 Console.Write("Содержимое массива nums2\n"
     + "после присваивания: \n");
 for (i = 0; i < 10; i++)
     Console.Write(nums2[i] + " ");
 Console.WriteLine();
 // Далее оперировать массивом nums1 посредством
 // переменной ссылки на массив nums2.
 nums2[3] = 73;
 Console.Write("Содержимое массива nums1 после изменения\n"
     + "посредством переменной nums2: ");
 for (i = 0; i < 10; i++)
     Console.Write(nums1[i] + " ");
 Console.WriteLine();
}
```

```
Содержимое массива nums
после присваивания:
0 1 2 3 4 5 6 7 8 9

Содержимое массива nums
после изменения
посредством переменной
nums2:
0 1 2 73 4 5 6 7 8 9
```

Применение свойства Length

```
static void Main()
{
    int[] nums = new int[10];
    Console.WriteLine("Длина массива nums равна " + nums.Length);
    // Использовать свойство Length для инициализации массива nums.
    for (int i = 0; i < nums.Length; i++)
        nums[i] = i * i;
    // А теперь воспользуемся свойством Length
    // для вывода содержимого массива nums.
    Console.Write("Содержимое массива nums: ");
    for (int i = 0; i < nums.Length; i++)
        Console.Write(nums[i] + " ");
    Console.WriteLine();
}
```

Результат:

```
Длина массива nums равна 10
Содержимое массива nums:  0 1 4 9 16 25 36 49 64 81
```

Свойство Length и метод GetLength()

```
static void Main(string[] args)
{
    int[, ,] nums = new int[10, 5, 6];
    Console.WriteLine("Длина массива nums равна " + nums.Length);
    Console.WriteLine("Количество строк: " + nums.GetLength(0));
    Console.WriteLine("Количество столбцов: " + nums.GetLength(1));
    Console.WriteLine("Кол-во элементов по 3му измерению: "
        + nums.GetLength(2));
}
```

Результат:

```
Длина массива nums равна 300
Количество строк: 10
Количество столбцов: 5
Количество элементов по 3му измерению: 6
```

Применение свойства Length

```
static void Main() {
    int i, j;
    int[] nums1 = new int[10]; int[] nums2 = new int[10];
    for (i = 0; i < nums1.Length; i++) nums1[i] = i;
    Console.Write("Исходное содержимое массива: ");
    for (i = 0; i < nums1.Length; i++)
        Console.Write(nums1[i]);
    Console.WriteLine();
   // проверить, достаточно ли длины массива nums2
    if (nums2.Length >= nums1.Length)
   // Скопировать элементы массива nums1
   //в массив nums2 в обратном порядке
        for (i=0, j=nums1.Length-1; i<nums1.Length; i++, j--)
            nums2[j] = nums1[i];
    Console.Write("Содержимое массива в обратном порядке: ");
    for (i = 0; i < nums2.Length; i++)
        Console.Write(nums2[i]);
    Console.WriteLine();
}
```

Результат:

```
Исходное содержимое массива: 0123456789
Содержимое массива в обратном порядке: 9876543210
```

Применение свойства Length к ступенчатым массивам

```
static void Main() {
    int[][] network_nodes = new int[4][];
    network_nodes[0] = new int[3];
    network_nodes[1] = new int[7];
    network_nodes[2] = new int[2];
    network_nodes[3] = new int[5];
    int i, j;
    // Сфабриковать данные об использовании ЦП.
    for (i = 0; i < network_nodes.Length; i++)
        for (j = 0; j < network_nodes[i].Length; j++)
            network_nodes[i][j] = i * j + 70;
    Console.WriteLine("Общее количество узлов сети: " +
    network_nodes.Length + "\n");
    for (i = 0; i < network_nodes.Length; i++) {
        for (j = 0; j < network_nodes[i].Length; j++) {
            Console.Write("Использование в узле сети " + i
                 + " ЦП " + j + ": ");
            Console.Write(network_nodes[i][j] + "% ");
            Console.WriteLine();
        }
        Console.WriteLine();
    }}
```

Результат:

```
Общее количество узлов 4
Использование в узле 0 ЦП 0: 70%
Использование в узле 0 ЦП 1: 70%
Использование в узле 0 ЦП 2: 70%

Использование в узле 1 ЦП 0: 70%
Использование в узле 1 ЦП 1: 71%
Использование в узле 1 ЦП 2: 72%
Использование в узле 1 ЦП 3: 73%
Использование в узле 1 ЦП 4: 74%
Использование в узле 1 ЦП 5: 75%
Использование в узле 1 ЦП 6: 76%

Использование в узле 2 ЦП 0: 70%
Использование в узле 2 ЦП 1: 72%

Использование в узле 3 ЦП 0: 70%
Использование в узле 3 ЦП 1: 73%
Использование в узле 3 ЦП 2: 76%
Использование в узле 3 ЦП 3: 79%
Использование в узле 3 ЦП 4: 82%
```

Оператор цикла foreach

- общая форма оператора цикла foreach.

```
foreach (тип имя_переменной_цикла in коллекция) оператор;
```

- Здесь тип ***имя_переменной_цикла*** обозначает тип и имя переменной управления циклом, которая получает значение следующего элемента коллекции на каждом шаге выполнения цикла foreach. А ***коллекция*** обозначает циклически опрашиваемую коллекцию, которая здесь и далее представляет собой массив.

Оператор цикла foreach

```
static void Main() {
  int sum = 0;
  int[] nums = new int[10];
  // Задать первоначальные значения
  // элементов массива nums.
  for (int i = 0; i < 10; i++)
  nums[i] = i;
  // Использовать цикл foreach для вывода значений
  // элементов массива и подсчета их суммы,
  foreach (int x in nums) {
    Console.WriteLine("Значение элемента равно: "+x);
    sum += x;
  }
  Console.WriteLine("Сумма равна: " + sum);
 }
```

Результат:

Значение элемента равно: 0
Значение элемента равно: 1
Значение элемента равно: 2
Значение элемента равно: 3
Значение элемента равно: 4
Значение элемента равно: 5
Значение элемента равно: 6
Значение элемента равно: 7
Значение элемента равно: 8
Значение элемента равно: 9
Сумма равна: 45

Оператор цикла foreach, преждевременное завершение

```
static void Main() {
    int sum = 0;
    int[] nums = new int[10];
    // Задать первоначальные значения элементов массива nums.
    for (int i = 0; i < 10; i++)
        nums[i] = i;
    // Использовать цикл foreach для вывода значений
    // элементов массива и подсчета их суммы.
    foreach (int x in nums)
    {
        Console.WriteLine("Значение элемента равно: "
                                                + x);
        sum += x;
        if (x == 4) break; // прервать цикл, как только
                // индекс массива достигнет 4
    }
    Console.WriteLine("Сумма первых 5 элементов: " + sum);
}
```

Результат:

Значение элемента равно: 0
Значение элемента равно: 1
Значение элемента равно: 2
Значение элемента равно: 3
Значение элемента равно: 4
Сумма первых 5 элементов: 10

Оператор цикла foreach, обращение к двумерному массиву

```
static void Main()
{
    int sum = 0;
    int[,] nums = new int[3, 5];
    // Задать первоначальные значения элементов массива nums.
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 5; j++)
            nums[i, j] = (i + 1) * (j + 1);
    // Использовать цикл foreach для вывода значений
    // элементов массива и подсчета их суммы.
    foreach (int x in nums)
    {
        Console.WriteLine("Значение элемента равно: " + x);
        sum += x;
    }
    Console.WriteLine("Сумма равна: " + sum);
}
```

Результат:

Значение элемента равно: 1
Значение элемента равно: 2
Значение элемента равно: 3
Значение элемента равно: 4
Значение элемента равно: 5
Значение элемента равно: 2
Значение элемента равно: 4
Значение элемента равно: 6
Значение элемента равно: 8
Значение элемента равно: 10
Значение элемента равно: 3
Значение элемента равно: 6
Значение элемента равно: 9
Значение элемента равно: 12
Значение элемента равно: 15
Сумма равна: 90

Оператор цикла foreach, поиск в массиве

```
static void Main() {
    int[] nums = new int[10];
    int val;
    bool found = false;  // Используется переменная-флаг
    // Задать первоначальные значения элементов массива nums.
    for (int i = 0; i < 10; i++)
        nums[i] = i;
    val = 5; // будем искать значение 5 в массиве
    // Использовать цикл foreach для поиска заданного
    // значения в массиве nums.
    foreach (int x in nums) {
        if (x == val) {
            found = true;
            break;
        }
    }
    if (found)
        Console.WriteLine("Значение найдено!");
}
```

Результат:
Значение найдено!

Строки

- Переменной ссылки на строку str присваивается ссылка на строковый литерал:

```
string str = "Строки в C# весьма эффективны.";
```

В данном случае переменная str инициализируется последовательностью символов "Строки в C# весьма эффективны.".

- Объект типа string можно также создать из массива типа char. Например:

```
char[] charray = {'t', 'e', 's', 't'} ;
string str = new string(charray);
```

Строки

Создать и вывести символьную строку

```
char[] charray = { 'Э', 'т', 'о', ' ', 'с', 'т', 'р', 'о', 'к', 'а', ' ' };
string str1 = new string(charray);
string str2 = "Еще одна строка.";
Console.WriteLine(str1);
Console.WriteLine(str2);
```

Результат

```
Это строка.
Еще одна строка.
```

Обращение со строками

Метод	Описание
`static int Compare (string strA, string strB, StringComparison comparisonType)`	Возвращает отрицательное значение, если строка strA меньше строки strB; положительное значение, если строка strA больше строки strB; и нуль, если сравниваемые строки равны. Способ сравнения определяется аргументом comparisonType
`bool Equals (string value, StringComparison comparisonType)`	Возвращает логическое значение true, если вызывающая строка имеет такое же значение, как и у аргумента value. Способ сравнения определяется аргументом comparisonType
`int IndexOf (char value)`	Осуществляет поиск в вызывающей строке первого вхождения символа, определяемого аргументом value. Применяется порядковый способ поиска. Возвращает индекс первого совпадения с искомым символом или -1, если он не обнаружен
`int IndexOf (string value, StringComparison comparisonType)`	Осуществляет поиск в вызывающей строке первого вхождения подстроки, определяемой аргументом value. Возвращает индекс первого совпадения с искомой подстрокой или -1, если она не обнаружена. Способ поиска определяется аргументом comparisonType
`int LastIndexOf (char value)`	Осуществляет поиск в вызывающей строке последнего вхождения символа, определяемого аргументом value. Применяется порядковый способ поиска. Возвращает индекс последнего совпадения с искомым символом или -1, если он не обнаружен
`int LastIndexOf (string value, StringComparison comparisonType)`	Осуществляет поиск в вызывающей строке последнего вхождения подстроки, определяемой аргументом value. Возвращает индекс последнего совпадения с искомой подстрокой или -1, если она не обнаружена. Способ поиска определяется аргументом comparisonType
`string ToLower (CultureInfo.CurrentCulture culture)`	Возвращает вариант вызывающей строки в нижнем регистре. Способ преобразования определяется аргументом culture
`string ToUpper (CultureInfo.CurrentCulture culture)`	Возвращает вариант вызывающей строки в верхнем регистре. Способ преобразования определяется аргументом culture

Обращение со строками

- Объекты типа string содержат также свойство Length, где хранится длина строки.
- метод Compare () вызывается по имени своего класса, а не по его экземпляру, общая форма:

```
результат = string.Compare(str1, str2, способ);
```

где способ обозначает конкретный подход к сравнению символьных строк.

- Отдельный символ выбирается из строки с помощью индекса, как в приведенном ниже фрагменте кода.

```
string str = "тест";
Console.WriteLine(str[1]);
```

В этом фрагменте кода выводится символ “е", который является первым в строке "тест".

Обращение со строками

- Для проверки двух строк на равенство служит оператор «**==**» .
- Для проверки двух строк на равенство с учетом культурной среды служит метод **Equals ()**, где непременно нужно указать способ сравнения в виде аргумента **StringComparison.CurrentCulture**.
- Следует также иметь в виду, что метод **Compare ()** служит для сравнения строк с целью определить отношение порядка, например для сортировки.

```
result = string.Compare(str1, str3,
StringComparison.CurrentCulture);
```

- С помощью оператора «+» можно сцепить (т.е. объединить вместе) две строки. Например:

```
string str1 = "Один";
string str2 = "Два";
string str3 = "Три";
string str4 = str1 + str2 + str3;
```

переменная str4 инициализируется строкой "ОдинДваТри".

Коды русских символов в Unicode

а 1072	с 1089
б 1073	т 1090
в 1074	у 1091
г 1075	ф 1092
д 1076	х 1093
е 1077	ц 1094
ж 1078	ч 1095
з 1079	ш 1096
и 1080	щ 1097
й 1081	ъ 1098
к 1082	ы 1099
л 1083	ь 1100
м 1084	э 1101
н 1085	ю 1102
о 1086	я 1103
п 1087	? 1104
р 1088	ё 1105

Операции со строками

```
using System.Globalization;
...
static void Main() {
   string str1 = "Программировать в .NET лучше всего на C#.";
   string str2 = "Программировать в .NET лучше всего на C#.";
   string str3 = "Строки в C# весьма эффективны.";
   string strUp, strLow;  int result, idx;
   Console.WriteLine("str1: " + str1);
   Console.WriteLine("Длина строки str1: " + str1.Length);
   // Создать варианты строки str1, набранные
   // прописными и строчными буквами.
   strLow = str1.ToLower(CultureInfo.CurrentCulture);
   strUp = str1.ToUpper(CultureInfo.CurrentCulture);
   Console.WriteLine("Вариант строки str1, " +
   "набранный строчными буквами:\n " + strLow);
   Console.WriteLine("Вариант строки str1, " +
   "набранный прописными буквами:\n " + strUp);
   Console.WriteLine();
   // Вывести строку str1 посимвольно.
   Console.WriteLine("Вывод строки str1 посимвольно.");
   for (int i = 0; i < str1.Length; i++)
       Console.Write(str1[i]);
```

Результат:

str1: Программировать в .NET лучше всего на C#.
Длина строки str1: 41

Вариант строки str1, набранный строчными буквами:
программировать в .net лучше всего на c#.

Вариант строки str1, набранный прописными буквами:
ПРОГРАММИРОВАТЬ В .NET ЛУЧШЕ ВСЕГО НА C#.

Вывод строки str1 посимвольно.
Программировать в .NET лучше всего на C#.

```
 // Сравнить строки способом порядкового сравнения.
if (str1 == str2)  Console.WriteLine("str1 == str2");
else  Console.WriteLine("str1 != str2");
if (str1 == str3) Console.WriteLine("str1 == str3");
else Console.WriteLine("str1 != str3");
// Сравнить строки с учетом культурной среды.
result = string.Compare(str3, str1, StringComparison.CurrentCulture);
if (result == 0)  Console.WriteLine("Строки str1 и str3 равны");
else if (result<0) Console.WriteLine("Строка str1 меньше строки str3");
else Console.WriteLine("Строка str1 больше строки str3");
Console.WriteLine();
// Присвоить новую строку переменной str2.
str2 = "Один Два Три Один";
// Поиск подстроки.
idx = str2.IndexOf("Один", StringComparison.Ordinal);
Console.WriteLine("Индекс первого вхождения"
                  + "подстроки <Один>: " + idx);
 idx = str2.LastIndexOf("Один",
                   StringComparison.Ordinal);
Console.WriteLine("Индекс последнего вхождения "
                           + " подстроки <Один>: " + idx);
}
```

Результат:

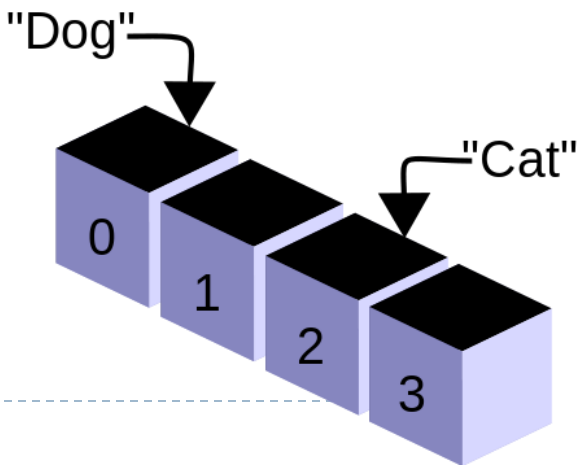
```
str1 == str2
str1 != str3
Строка str1 больше
строки str3
Индекс первого вхождения
подстроки <Один>: 0
Индекс последнего
вхождения подстроки
<Один>: 13
```

Массивы строк

```
static void Main() {
    string[] str = { "Это", "очень",
          "простой", "тест." };

    Console.WriteLine("Исходный массив строк: ");
    for (int i = 0; i < str.Length; i++)
        Console.Write(str[i] + " ");
    Console.WriteLine();

    // Изменить строку.
    str[1] = "тоже";
    str[3] = "до предела тест !";

    Console.WriteLine("Видоизмененный массив: ");
    for (int i = 0; i < str.Length; i++)
        Console.Write(str[i] + " ");
}
```

Результат:

Исходный массив строк:
Это очень простой тест.
Видоизмененный массив:
Это тоже простой до предела тест!




Эй, кот. Уже час прошел.
Ты там живой?
Барашек, отвали. не
смешно...
BAD

Вывод отдельных цифр целого числа словами

```
static void Main() {
    int num, nextdigit, numdigits;
    int[] n = new int[20];
    string[] digits = {"ноль", "один", "два",
    "три", "четыре", "пять", "шесть", "семь", "восемь", "девять" };
    num = 1908;
    Console.WriteLine("Число: " + num);
    Console.Write("Число словами: ");
    nextdigit = 0; numdigits = 0;
    // Получить отдельные цифры и сохранить их в массиве n.
    // Эти цифры сохраняются в обратном порядке
    do {
        nextdigit = num % 10;
        n[numdigits] = nextdigit;
        numdigits++;
        num = num / 10;
    } while (num > 0);
    // см. дальше...
```

Результат:

```
Число: 1908
Число словами: один
девять ноль восемь
```

Вывод отдельных цифр целого числа словами

```
    // ...продолжение:
    // Теперь в массиве n хранятся значения {8, 0, 9, 1}
    numdigits--; // numdigits==3 (индекс последнего элемента)
   // Вывести полученные слова.
    for (; numdigits >= 0; numdigits--)
        Console.Write(digits[n[numdigits]] + " ");
    Console.WriteLine();
}
```

Результат:

```
Число: 1908
Число словами: один
девять ноль восемь
```

Применение строк в операторах switch

```
static void Main() {
    string[] strs = { "один", "два", "три", "два", "один" };
    foreach (string s in strs) {
        switch (s) {
            case "один":
                Console.Write(1);
            case "два":
                Console.Write(2);
                break;
            case "три":
                Console.Write(3);
                break;
        }
    }
    Console.WriteLine();
}
```

Результат:
12321

Постоянство строк

- Содержимое объекта типа string **не подлежит изменению**. Это означает, что однажды созданную последовательность символов изменить нельзя.
- форма метода **Substring ()**:

```
string Substring(int индекс_начала, int длина)
```

- где **индекс_начала** обозначает начальный индекс исходной строки, а **длина** — длину выбираемой подстроки.
- Пример:

```
static void Main() {
    string orgstr = "В C# упрощается обращение со строками.";
    // сформировать подстроку
    string substr = orgstr.Substring(5, 20);
    Console.WriteLine("orgstr: " + orgstr);
    Console.WriteLine("substr: " + substr);
}
```

- Результат:

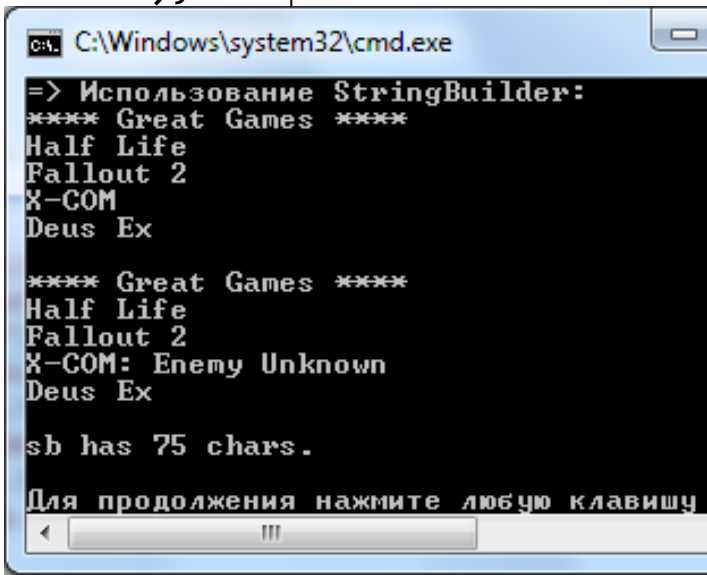
```
orgstr: В C# упрощается обращение со строками,
substr: упрощается обращение
```

Изменяемые строки – StringBuilder

```
using System.Text;
```

- Класс **StringBuilder** имеет два главных свойства:
  - ***Length***, показывающее длину строки, содержащуюся в объекте в данный момент
  - ***Capacity***, указывающее максимальную длину строки, которая может поместиться в выделенную для объекта память

```
static void Main()
{
    Console.WriteLine("=> Использование StringBuilder:");
    StringBuilder sb = new StringBuilder("**** Great Games ****");
    sb.Append("\n");
    sb.AppendLine("Half Life");
    sb.AppendLine("Fallout 2");
    sb.AppendLine("X-COM");
    sb.AppendLine("Deus Ex");
    Console.WriteLine(sb.ToString());
    sb.Replace("COM", "COM: Enemy Unknown");
    Console.WriteLine(sb.ToString());
    Console.WriteLine("sb has {0} chars.", sb.Length);
    Console.WriteLine();
}
```



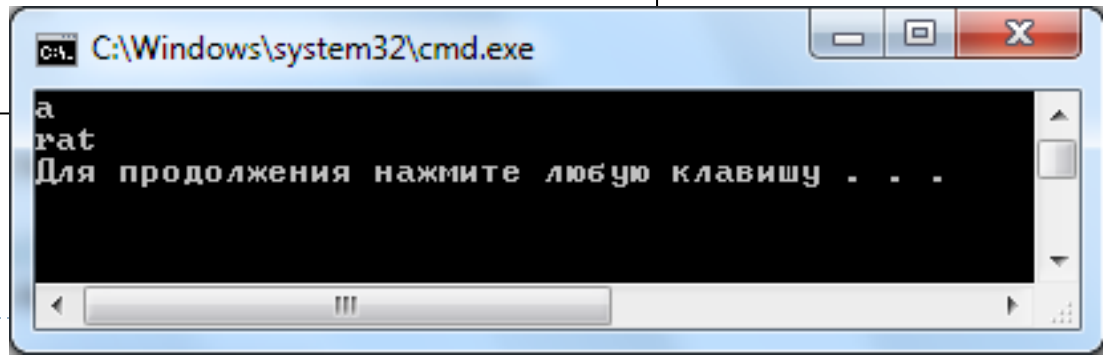
Методы класса StringBuilder

Метод	Описание
Append()	Добавляет строку к текущей строке
AppendFormat()	Добавляет строку, сформированную в соответствии со спецификатором формата
Insert()	Вставляет подстроку в строку
Remove()	Удаляет символ из текущей строки
Replace()	Заменяет все вхождения символа другим символом или вхождения подстроки другой подстрокой
ToString()	Возвращает текущую строку в виде объекта System.String (переопределение метода класса System.Object)

StringBuilder

- Изменение отдельных символов в строке

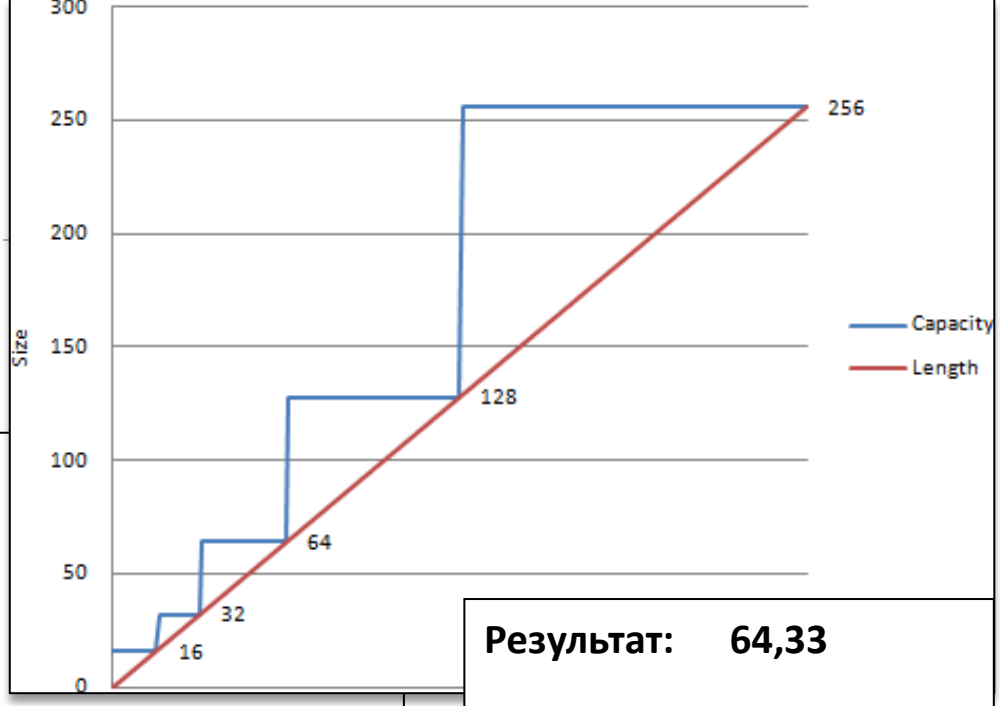
```
static void Main(){
    StringBuilder builder = new StringBuilder();
    builder.Append("cat");

    // Вывести вторую букву
    Console.WriteLine(builder[1]);

    // Заменить первую букву
    builder[0] = 'r';
    Console.WriteLine(builder.ToString());
}
```



StringBuilder

- Длина (Length)

  и вместимость (Capacity)




```
using System.IO;
using System.Text;

class Program {
  static void Main() {
    using (var writer = new StreamWriter("data.txt")){
      StringBuilder builder = new StringBuilder();
      for (int i = 0; i <= 256; i++) {
        writer.Write(builder.Capacity);
        writer.Write(",");
        writer.Write(builder.Length);
        writer.WriteLine();
        builder.Append("1"); //Добавить один символ
      }
    }
  }
}
```

Результат:

16,0	64,33
16,1	…
16,2	64,64
…	128,65
16,16	…
32,17	128,128
32,18	256,129
…	…
32,32	256,256