

# C#

---

---

## «Символы»:

1. Символы. Класс Char.
2. Некоторые методы для работы с символами.
3. Строки. Операции над строками.
4. Класс String. Некоторые свойства и методы для работы со строками.

# Символы. Класс Char

Каждый символ занимает два байта и использует кодировку Unicode.

Описание с одновременной инициализацией:

**//просто символ**

**char ch1='А';**

**//символ в виде шестнадцатеричного кода**

**char ch1='\xF';**

**//символ в виде escape-**

**//последовательности Unicode**

**char ch1='\uA81B';**

Для символьных констант используются одинарные кавычки.

# Символы. Класс Char

Тип `char`, как и все типы C#, является классом. Поэтому переменную `ch` можно объявлять в объектном стиле, используя `new` и вызов конструктора класса без параметров.

```
char ch = new char();
```

```
ch = 'Z';
```

# Символы. Класс Char.

## Получение кода символа

Символы кодируются в Unicode. Получить Unicode символа можно так:

```
char c = '5';  
int k = (int)c; Console.WriteLine(k);
```

Выведет 53.

Для английских букв и цифр кодировка Unicode совпадает с кодировкой ASCII.

Для русских букв нет.

# Символы. Класс Char.

## Получение символа по его коду

Получить символ по его Unicode коду можно так:

```
int k = 100;
```

```
char c = (char)k; Console.WriteLine(c);
```

Выведет d.

```
int k = 1046;
```

```
char c = (char)k; Console.WriteLine(c);
```

Выведет Ж.



# Класс Char. Метод IsDigit для работы с символами

Метод **IsDigit** возвращает true, если символ является десятичной цифрой и false в противном случае.

Например:

```
char ch = new char(); ch = '5';  
bool p2 = char.IsDigit(ch);  
    if (p2) Console.WriteLine(" Цифра ");  
    else Console.WriteLine("Не цифра");
```

Такой фрагмент программы выведет на экран слово «Цифра».

## Класс Char. Метод IsLetter для работы с символами

Метод **IsLetter** возвращает true, если символ является буквой (любой) и false в противном случае.

Например:

```
char ch1 = 'Я';  
bool p1 = char.IsLetter(ch1);  
    if (p1) Console.WriteLine("Буква ");  
    else Console.WriteLine("Не буква");
```

Такой фрагмент программы выведет слово «Буква».



## Класс Char. Метод IsControl для работы с символами

Метод IsControl возвращает true, если символ является управляющим и false в противном случае.

Например:

```
char ch3 = '\xA';  
bool p3 = char.IsControl(ch3);  
if (p3)  
    Console.WriteLine(" Управляющий");  
else  
    Console.WriteLine("Не управляющий");
```

Такой фрагмент программы выведет слово «Управляющий».

## Строки. Описание

Тип `string` задает строки переменной длины.

Описание с одновременной инициализацией:

```
string world = "Мир";
```

Для строковых констант используются двойные кавычки.

# Операции над строками

- Над строками определены следующие операции:
- присваивание (=);
- две операции проверки эквивалентности (==) и (!=);
- конкатенация или сцепление строк (+);
- Выбор символа по индексу ([ ]).  
Нумерация символов с 0. Только для чтения!

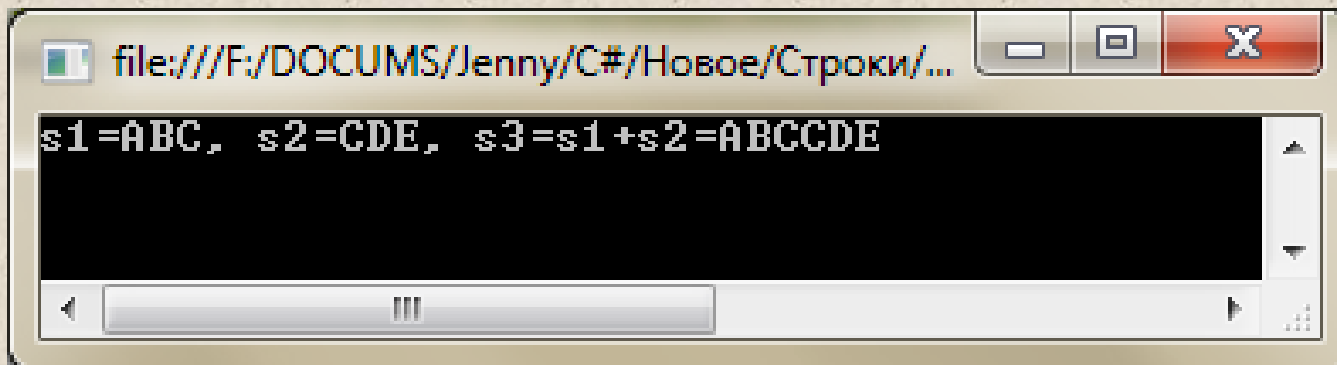
# Сложение

//операции над строками

```
string s1 = "ABC", s2 = "CDE";
```

```
string s3 = s1 + s2;
```

```
Console.WriteLine("s1={0}, s2={1}, s3=s1+s2={2}", s1,  
s2, s3);
```



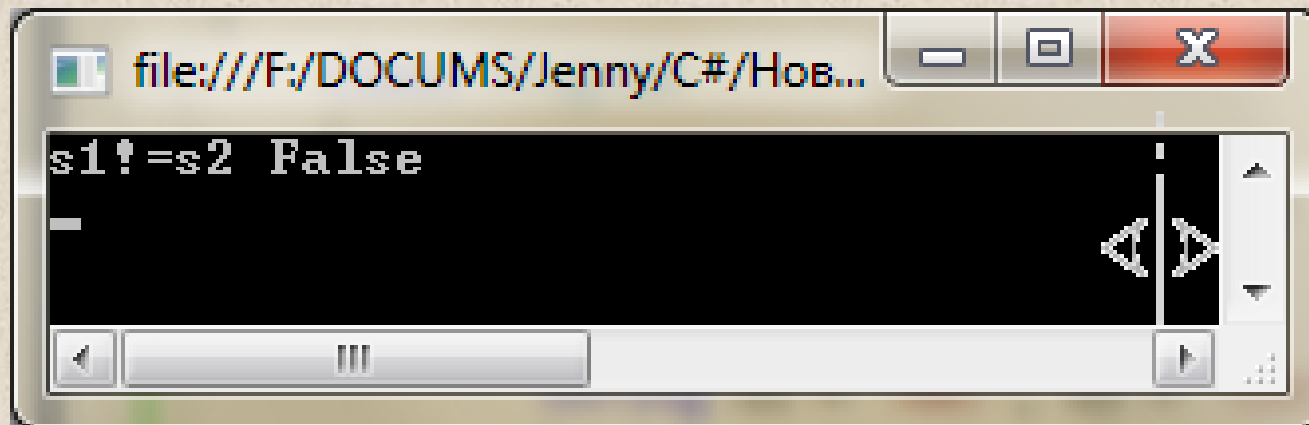
## Сравнение

//операции над строками

```
string s1 = "ABC", s2 = "CDE";
```

```
bool b1 = (s1 == s2);
```

```
Console.WriteLine("s1!=s2 {0}", b1);
```





## Выбор символа по индексу

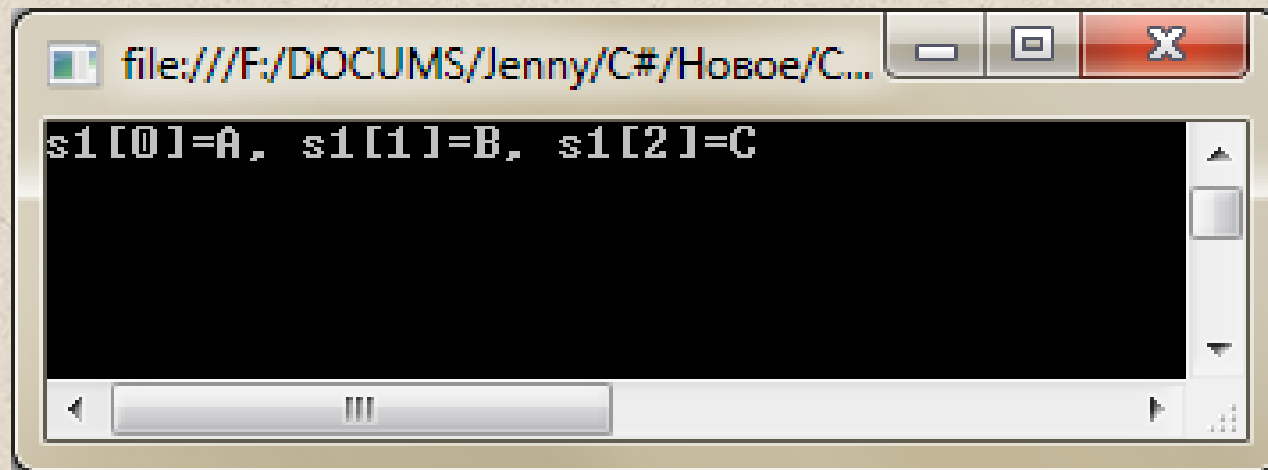
//операции над строками

```
string s1 = "ABC";
```

```
char ch1,ch2,ch3;
```

```
ch1 = s1[0]; ch2 = s1[1]; ch3 = s1[2];
```

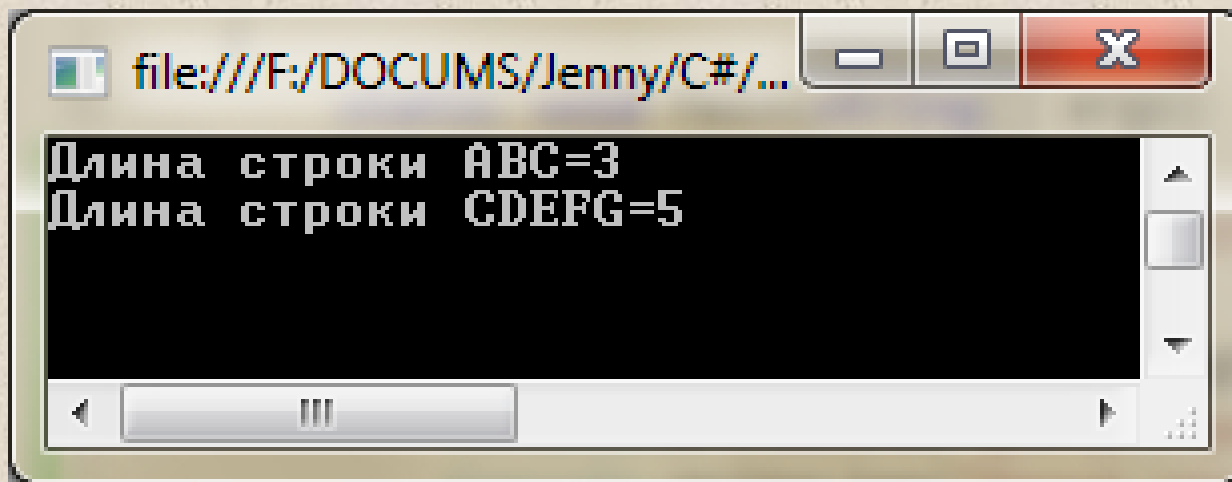
```
Console.WriteLine("s1[0]={0}, s1[1]={1}, s1[2]={2}",  
ch1, ch2, ch3);
```



# Определение длины строки.

## Свойство Length

```
string s1 = "ABC", s2 = "CDEFG";  
int L1,L2;  
L1 = s1.Length; L2 = s2.Length;  
Console.WriteLine("Длина строки {0}={1}",s1, L1);  
Console.WriteLine("Длина строки {0}={1}", s2, L2);
```



# Неизменяемый класс String

Обращаться к отдельному элементу строки по индексу можно только для получения значения, но не для его изменения. Это связано с тем, что строки типа `string` относятся к так называемым неизменяемым типам данных.

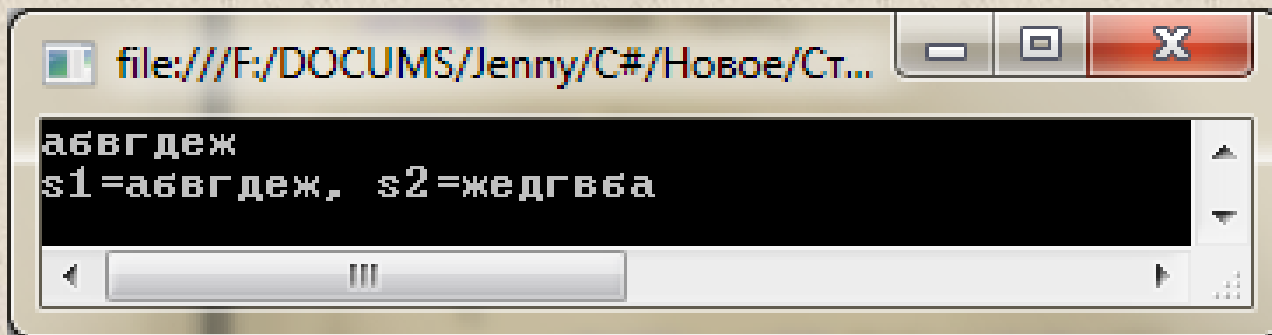
```
s1= "Zenon"; s1[0]='L';// будет ошибка
```

Методы, изменяющие содержимое строки, на самом деле создают новую копию строки.

## Некоторые особенности

В С# сохранилась возможность работать со строкой как с массивом символов.

В следующем фрагменте программы с клавиатуры вводится строка `s1` и формируется строка `s2`, содержащая символы строки `s1` в обратном порядке:

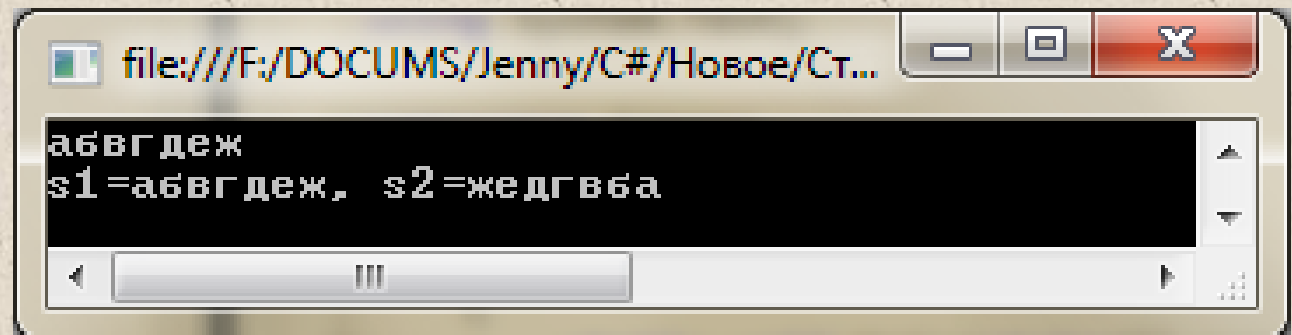


The screenshot shows a Notepad window with the following text:

```
file:///F:/DOCUMS/Jenny/C#/Новое/Ст...  
абвгдеж  
s1=абвгдеж, s2=жедгвба
```

The window title bar indicates the file path: `file:///F:/DOCUMS/Jenny/C#/Новое/Ст...`. The text area contains two lines: the first line is `абвгдеж`, and the second line is `s1=абвгдеж, s2=жедгвба`. The window has standard Windows controls (minimize, maximize, close) and a scrollbar.

```
string s1, s2;  
s1 = Console.ReadLine();  
s2 = "";  
// i параметр цикла, L длина строки s1  
int i, L;  
L = s1.Length;  
//накопление строки s2  
for (i = 0; i < L; i++)  
s2 = s2+s1[L - i - 1];  
Console.WriteLine("s1={0}, s2={1}",s1,s2);  
Console.ReadKey();
```



```
file:///F:/DOCU...  
абвгдеж  
s1=абвгдеж, s2=жедгвба
```

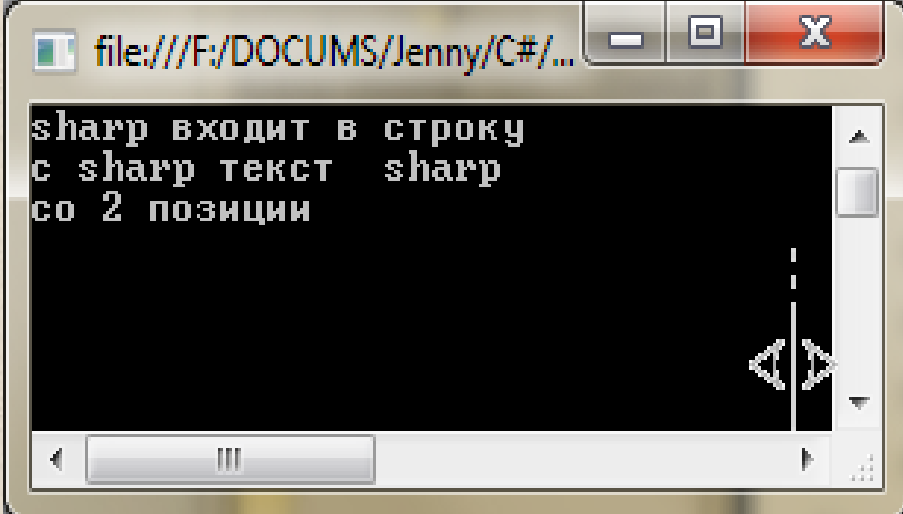


# Класс `string` . Метод `IndexOf`

Метод **`IndexOf`** используется для определения индекса первого вхождения подстроки в указанную строку. Например:

```
string stroka1 = "c sharp текст  sharp ";  
string stroka2 = "sharp";  
int i = stroka1.IndexOf(stroka2);  
if (i >=0) Console.WriteLine(stroka2 +  
" входит в строку \n" + stroka1 +  
" \n со "+i+" позиции")
```

Такой фрагмент программы выведет на экран:



The screenshot shows a Windows console window with a title bar containing the file path `file:///F:/DOCU.../Jenny/C#...`. The console output is as follows:

```
sharp входит в строку  
c sharp текст  sharp  
со 2 позиции
```

# Извлечение подстрок

Метод **Substring** используется для извлечения подстроки из строки. Извлеченное значение присваивается новой строке. Например:

```
string s3 = "Visual C# Express";  
string s4 = s3.Substring(7, 2);  
Console.WriteLine(s4);
```

Такой фрагмент программы выведет на экран: **C#**

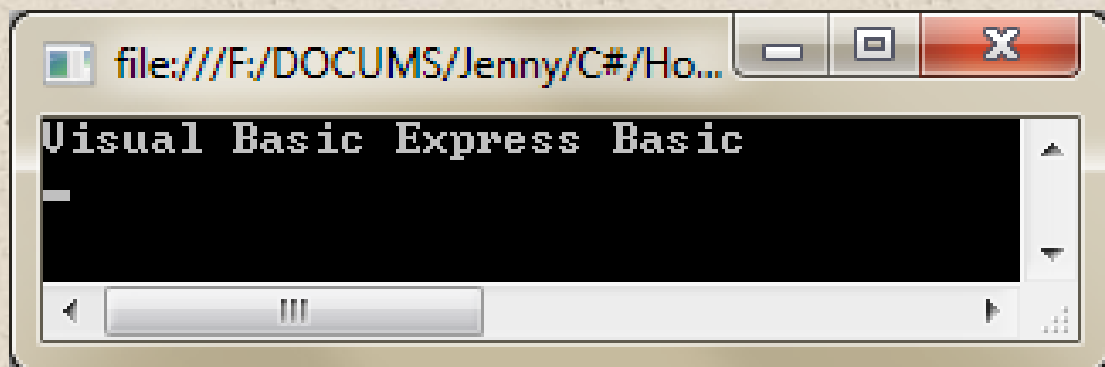
# Замена по образцу

Метод **Replace** используется для замены всех вхождений подстроки в строку по образцу.

Например:

```
string s3 = "Visual C# Express C#";  
string s5 = s3.Replace("C#", "Basic");  
Console.WriteLine(s5);
```

Такой фрагмент программы выведет на экран:



# Удаление фрагментов и вставка строк в строки

Может быть выполнена с помощью методов **Remove** и **Insert**:

```
string x = "ZX Spectrum";
```

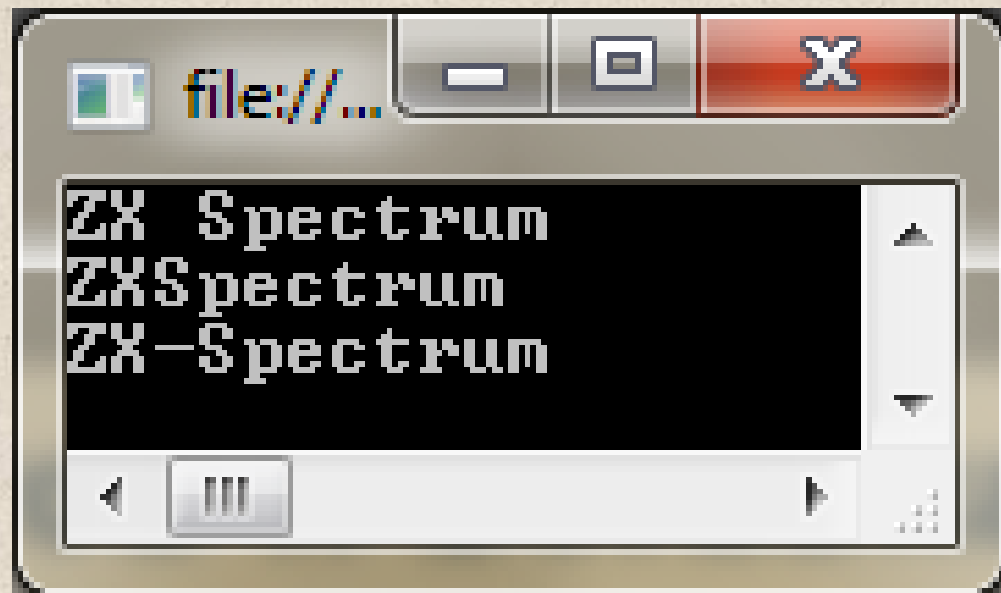
```
Console.WriteLine(x);
```

```
x = x.Remove(2, 1);
```

```
Console.WriteLine(x);
```

```
x = x.Insert(2, "-");
```

```
Console.WriteLine(x);
```

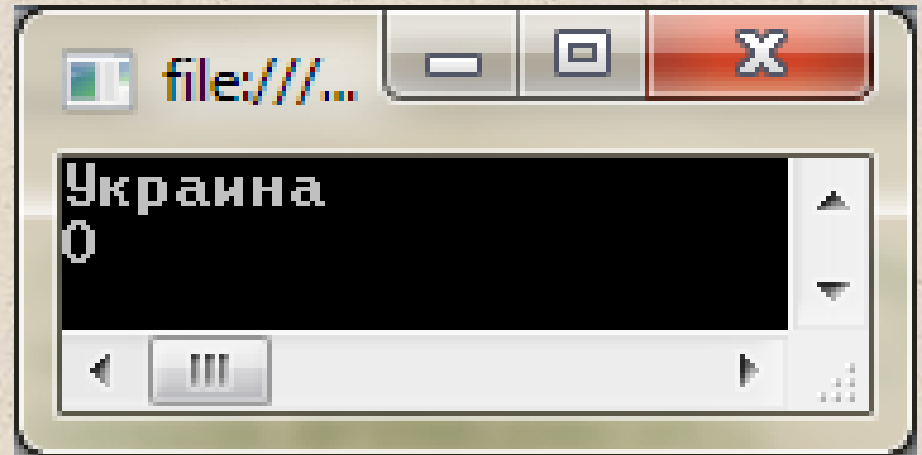


## Массивы из строк

Могут быть инициализированы начальными значениями:

```
string[] x =  
    { "Украина",  
      "Одесса",  
      "ул. Щепкина",  
      "д. 5"  
    };
```

```
Console.WriteLine(x[0]);  
Console.WriteLine(x[1][0]);
```





# Класс `string` . Метод `Join`

Метод `Join` позволяет соединить массив строк в единую строку. При конкатенации между элементами массива вставляются разделители. Например:

```
//задан массив строк из 3-х элементов
```

```
string [] a = {"qwer","tyu","123456"};
```

```
//в строке st элементы массива через запятую
```

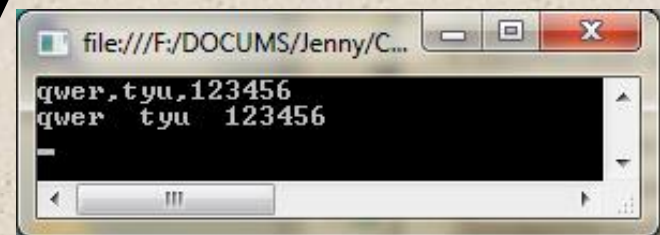
```
string st = string.Join(",", a);
```

```
Console.WriteLine(st);
```

```
//в строке st1 элементы массива через пробел
```

```
string st1 = string.Join(" ", a);
```

```
Console.WriteLine(st1);
```



## Класс `string` . Метод `Split`

Метод **`Split`** позволяет осуществить разбор текста на элементы. Метод применяется к строке. В качестве параметра указывается один или несколько разделителей. В результате применения метода получается строковый массив, содержащий элементы (слова) строки.

# Пример 1 (один разделитель – пробел)

```
// исходная строка, содержит слова через пробел  
string txt = "А это пшеница, которая в темном"+  
"чулане хранится," + " в доме, который построил"+  
"Джек!";
```

```
Console.WriteLine(txt);
```

```
// описание массива слов
```

```
string[] Ws;
```

```
//применение метода к строке. Разделитель пробел
```

```
Ws = txt.Split(' ');
```

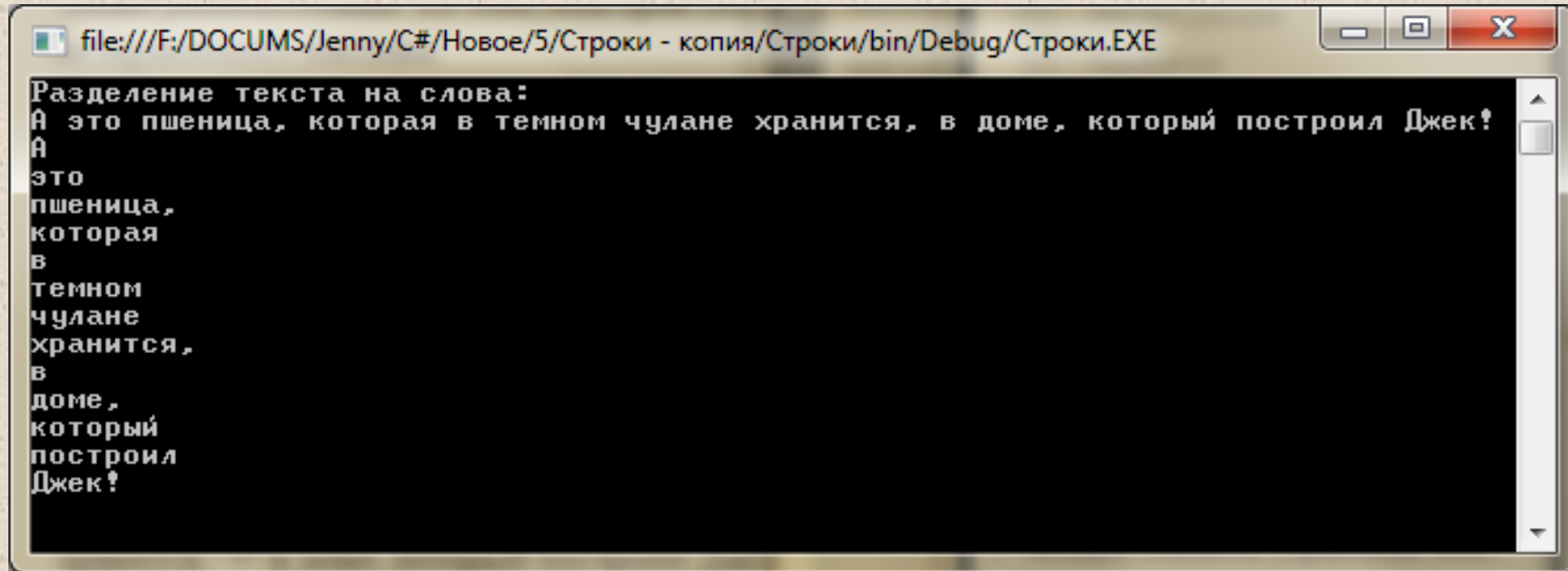
```
int i;
```

```
// СВОЙСТВО Ws.Length длина массива слов
```

```
for ( i = 0; i < Ws.Length; i++)
```

```
Console.WriteLine(Ws[i]);
```

# Пример 1 (один разделитель – пробел)



The screenshot shows a Windows command prompt window with the title bar "file:///F:/DOCUMS/Jenny/C#/Новое/5/Строки - копия/Строки/bin/Debug/Строки.EXE". The window has standard Windows window controls (minimize, maximize, close) on the right. The command prompt displays the following text:

```
Разделение текста на слова:  
А это пшеница, которая в темном чулане хранится, в доме, который построил Джек?  
А  
это  
пшеница,  
которая  
в  
темном  
чулане  
хранится,  
в  
доме,  
который  
построил  
Джек?
```

The text is split into words and punctuation marks across multiple lines, demonstrating the result of a word-splitting operation where spaces are used as delimiters.

Запятые относятся к словам

## Пример 2 (один разделитель – запятая)

// исходная строка

```
string txt = "А это пшеница, которая в темном"+  
"чулане хранится," + " в доме, который построил"+  
"Джек!";
```

```
Console.WriteLine(txt);
```

// описание массива слов

```
string[] Ws;
```

// применение метода к строке. Разделитель запятая

```
Ws = txt.Split(',', '');
```

```
int i;
```

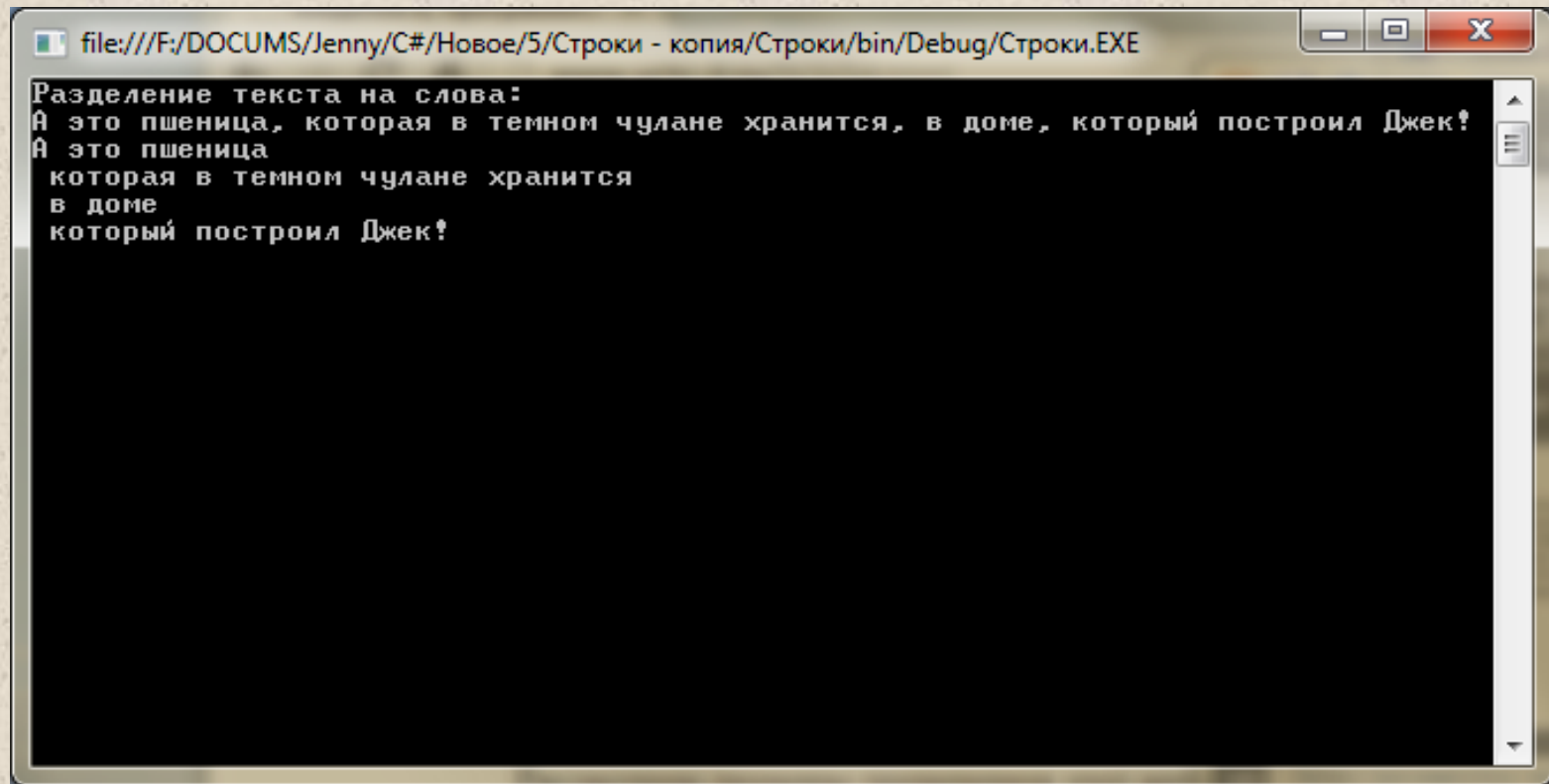
// вывод массива слов на экран

```
for ( i = 0; i < Ws.Length; i++)
```

```
Console.WriteLine(Ws[i]);
```



## Пример 2 (один разделитель – запятая)



The screenshot shows a Windows command prompt window with the title bar text: file:///F:/DOCUMS/Jenny/C#/Новое/5/Строки - копия/Строки/bin/Debug/Строки.EXE. The window contains the following text:

```
Разделение текста на слова:  
А это пшеница, которая в темном чулане хранится, в доме, который построил Джек!  
А это пшеница  
    которая в темном чулане хранится  
    в доме  
    который построил Джек!
```

В предложении 3 запятые. Предложение разбито на 4 части.

### Пример 3 (два разделителя – запятая и пробел)

// исходная строка

```
string txt = "А это пшеница, которая в темном"+  
"чулане хранится," + " в доме, который построил"+  
"Джек!";
```

```
Console.WriteLine(txt);
```

// описание массива слов

```
string[] Ws;
```

// применение метода к строке. Разделители запятая и пробел

```
Ws = txt.Split (',', ' ');
```

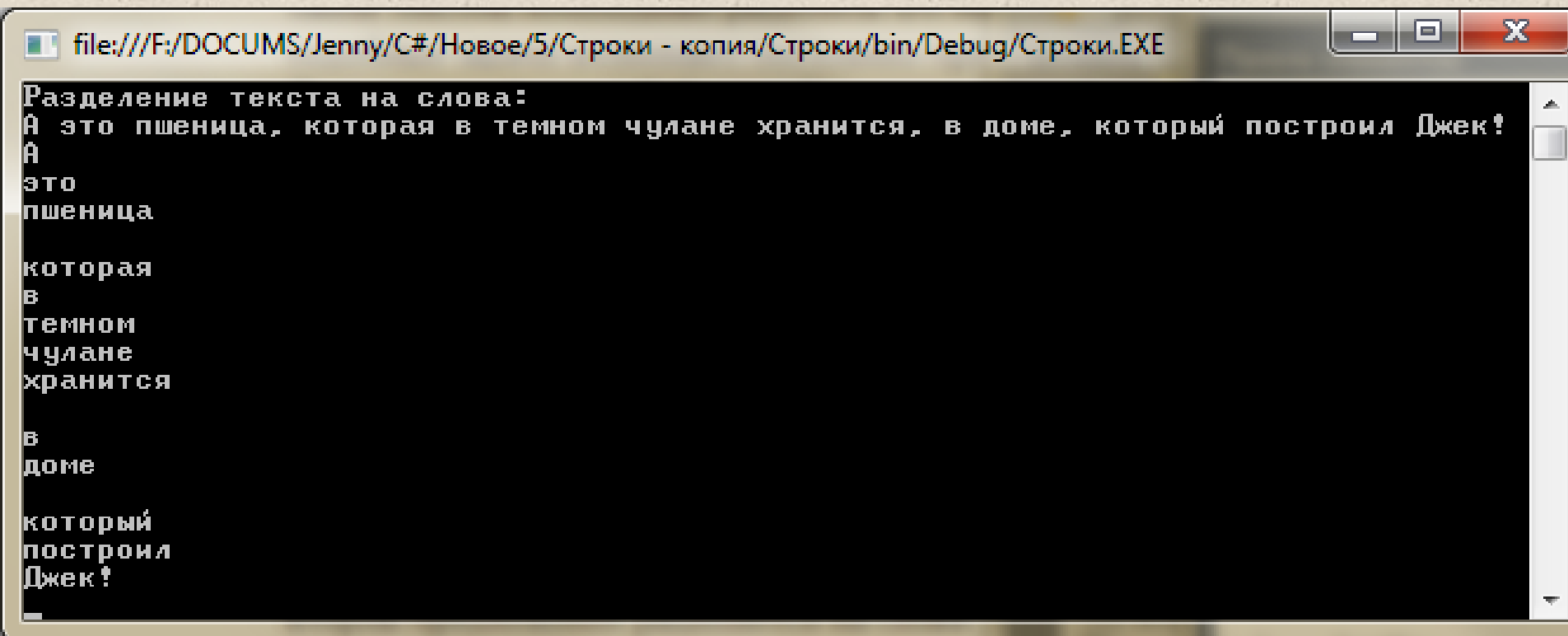
```
int i;
```

// вывод массива на экран

```
for ( i = 0; i < Ws.Length; i++)
```

```
Console.WriteLine(Ws[i]);
```

## Пример 3 (два разделителя – запятая и пробел)



The screenshot shows a Windows command prompt window with the title bar text: file:///F:/DOCUMS/Jenny/C#/Новое/5/Строки - копия/Строки/bin/Debug/Строки.EXE. The window has standard Windows window controls (minimize, maximize, close). The command prompt displays the following text:

```
Разделение текста на слова:  
А это пшеница, которая в темном чулане хранится, в доме, который построил Джек!  
А  
это  
пшеница  
  
которая  
в  
темном  
чулане  
хранится  
  
в  
доме  
  
который  
построил  
Джек!
```

В массиве появились пустые слова