Частное учреждение образования

«Колледж бизнеса и права»

|  |
| --- |
| УТВЕРЖДАЮ  Заведующий  методическим кабинетом  \_\_\_\_\_\_\_\_\_\_\_\_\_Е.В.Фалей  « \_\_\_\_ » \_\_\_\_\_\_\_\_\_ 2016 |

|  |  |
| --- | --- |
| Специальность: 2-40 01 01 «Программное обеспечение информационных технологий» | Учебная практика по программированию |
| Составлена в соответствии с учебной программой, утвержденной директором Колледжа бизнеса и права 31.05.2016 | |

**Практическая работа №6**

**Инструкционно-технологическая карта**

Тема: Классы Char, String, StringBuilder. Базовые операции со строками в .NET Framework.

Цель: Развить умения по использованию базовых операций со строками.

Время выполнения: 2 часа

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Ознакомиться с теоретическими сведениями.
2. Набрать, отладить, протестировать и выполнить примеры программ, представленные в теоретических сведениях.
3. Получить у преподавателя вариант задания и выполнить его.
4. Оформить решение в отчет.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

**Классы Char, String, StringBuilder**

Для работы со строками в языке C# существует несколько возможностей:

* используя символьный тип char(символ), который соответствует структуре System.Char, для организации строки – массив символов типа char, который соответствует классу System.Array;
* используя строки типа string;
* используя строки типа StringBuilder.

Символьный тип относится к встроенным типам данных C# и соответствует структуре System.Char, которая представляет символ как элемент кода UTF-16. В этой структуре определены статические методы, позволяющие сравнивать объекты, преобразовывать символы из кодировки UTF-16 в кодовую точку Юникода и наоборот, узнать или получить вид и категорию символа, а также преобразовать символ в верхний или нижний регистр и в число.

Примечание: .NET Framework использует Char структуры для представления символа Юникода. В стандарте Юникода идентифицируется каждый символ Юникода с уникальным 21-Битовым скалярным числом, называемым кодовой точкой и определяется форма кодировки UTF-16, указывающая, как кодируется кодовая точка в последовательность 16-разрядных значений одного или нескольких. Каждый 16-битовое значение лежит в диапазоне от шестнадцатеричное 0x0000 до 0xFFFF и хранится в структуре Char. Значением объекта типа Char является его значение 16-разрядное числовое (порядковый номер).

Класс String представляет текст как последовательность из частей кода UTF-16. [String](https://msdn.microsoft.com/ru-ru/library/system.string(v=vs.110).aspx) является неизменяемым типом.

Класс StringBuilder представляет объект строкового типа, значение которого является изменяемой последовательностью символов. Хотя [String](https://msdn.microsoft.com/ru-ru/library/system.string(v=vs.110).aspx) и StringBuilder оба представляют последовательность символов, они реализуются по-разному.

**Базовые операции со строками в .NET Framework**

К базовым операциям со строками в .NET Framework относятся:

* [создание новых строк в .NET Framework](https://msdn.microsoft.com/ru-ru/library/3b23dab9(v=vs.110).aspx);
* [сокращение и удаление знаков из строк в .NET Framework](https://msdn.microsoft.com/ru-ru/library/kxbw3kwc(v=vs.110).aspx);
* [заполнение строк в .NET Framework](https://msdn.microsoft.com/ru-ru/library/66f6d830(v=vs.110).aspx);
* [сравнение строк в .NET Framework](https://msdn.microsoft.com/ru-ru/library/fbh501kz(v=vs.110).aspx);
* [смена регистра в .NET Framework](https://msdn.microsoft.com/ru-ru/library/s4xaew6s(v=vs.110).aspx);
* [использование класса StringBuilder в .NET Framework](https://msdn.microsoft.com/ru-ru/library/2839d5h5(v=vs.110).aspx).

# Создание новых строк в .NET Framework

 .NET Framework позволяет создавать строки с помощью простой операции присваивания, а также перегружать конструктор класса для создания строк с помощью нескольких различных параметров. Кроме того, в классе [System.String](https://msdn.microsoft.com/ru-ru/library/system.string(v=vs.110).aspx) .NET Framework имеются методы для создания строковых объектов путем объединения нескольких строк, массивов строк или объектов.

## Создание строк с помощью присваивания. Самый простой способ создать новый объект [String](https://msdn.microsoft.com/ru-ru/library/system.string(v=vs.110).aspx) — присвоить строковый литерал объекту [String](https://msdn.microsoft.com/ru-ru/library/system.string(v=vs.110).aspx).

## Создание строк с помощью конструктора класса. С помощью перегруженных конструкторов класса [String](https://msdn.microsoft.com/ru-ru/library/system.string(v=vs.110).aspx) можно создавать строки из массивов знаков. Кроме того, строки можно создавать путем копирования того или иного знака указанное количество раз.

## Методы, возвращающие строки. В следующей таблице перечислено несколько полезных методов, которые возвращают строковые объекты.

| **Название метода** | **Применение** |
| --- | --- |
| [String.Format](https://msdn.microsoft.com/ru-ru/library/system.string.format(v=vs.110).aspx) | Создание форматированной строки из набора объектов ввода. |
| [String.Concat](https://msdn.microsoft.com/ru-ru/library/system.string.concat(v=vs.110).aspx) | Создание строк из двух и более строк. |
| [String.Join](https://msdn.microsoft.com/ru-ru/library/system.string.join(v=vs.110).aspx) | Создание новой строки с помощью объединения массива строк. |
| [String.Insert](https://msdn.microsoft.com/ru-ru/library/system.string.insert(v=vs.110).aspx) | Создание новой строки с помощью вставки строки в указанную позицию существующей строки. |
| [String.CopyTo](https://msdn.microsoft.com/ru-ru/library/system.string.copyto(v=vs.110).aspx) | Копирование указанных знаков в строке в указанную позицию в массиве знаков. |

## **Формат**

Метод String**.**Format используется для создания форматированных строк и соединения строк, представляющих несколько объектов. Этот метод автоматически преобразует в строку любой переданный объект. Например, если приложению необходимо отобразить для пользователя значение Int32и значение DateTime, легко создается строка для представления этих значений с помощью метода Format. В следующем примере метод Formatиспользуется для создания строки, содержащей целочисленную переменную.

int numberOfFleas = 12;

string miscInfo = String.Format("Your dog has {0} fleas. " +

"It is time to get a flea collar. " +

"The current universal date is: {1:u}.",

numberOfFleas, DateTime.Now);

Console.WriteLine(miscInfo);

// The example displays the following output:

// Your dog has 12 fleas. It is time to get a flea collar.

// The current universal date is: 2008-03-28 13:31:40Z.

В этом примере значение [DateTime.Now](https://msdn.microsoft.com/ru-ru/library/system.datetime.now(v=vs.110).aspx) отображает текущие дату и время в соответствии с языком и региональными параметрами, связанными с текущим потоком.

## **Concat**

Метод String.Concatиспользуется для простого создания нового объекта строки из двух или более существующих объектов. Он позволяет использовать независимый от языка способ сцепления строк. Этот метод принимает любой класс, производный от System.Object. В следующем примере создается строка из двух существующих объектов строки и знака разделения.

string helloString1 = "Hello";

string helloString2 = "World!";

Console.WriteLine(String.Concat(helloString1, ' ', helloString2));

// The example displays the following output:

// Hello World!

## **Join**

Метод String.Joinсоздает новую строку из массива строк и разделительной строки. Этот метод полезен в случае необходимости сцепления нескольких строк и создания списка, отделенного, например, запятой. В следующем примере используется пробел для привязки массива строк.

string[] words = {"Hello", "and", "welcome", "to", "my" , "world!"};

Console.WriteLine(String.Join(" ", words));

// The example displays the following output:

// Hello and welcome to my world!

## **Insert**

Метод String.Insertсоздает новую строку с помощью вставки строки в указанную позицию другой строки. Этот метод использует индекс с отсчетом от нуля. В следующем примере строка вставляется в пятую позицию индекса MyString, и создается новая строка с этим значением.

string sentence = "Once a time.";

Console.WriteLine(sentence.Insert(4, " upon"));

// The example displays the following output:

// Once upon a time.

## **CopyTo**

Метод String.CopyTo копирует часть строки в массив знаков. Можно указать начальный индекс строки и число копируемых знаков. Для копирования этим методом необходимы исходный индекс, массив знаков, индекс назначения и число знаков. Все индексы отсчитываются от нуля. В следующем примере метод CopyToиспользуется для копирования знаков слова "Hello" из объекта строки в первую позицию индекса массива знаков.

string greeting = "Hello World!";

char[] charArray = {'W','h','e','r','e'};

Console.WriteLine("The original character array: {0}", new string(charArray));

greeting.CopyTo(0, charArray,0 ,5);

Console.WriteLine("The new character array: {0}", new string(charArray));

// The example displays the following output:

// The original character array: Where

// The new character array: Hello

## **Сокращение и удаление знаков из строк в .NET Framework**

 При разборе предложения на отдельные слова может оказаться, что некоторые слова на обоих концах содержат пробелы. В этом случае для удаления ряда пропусков или других знаков из указанного места строки используются методы сокращения из класса System.String. В следующей таблице представлены доступные методы сокращения.

| **Название метода** | **Применение** |
| --- | --- |
| [String.Trim](https://msdn.microsoft.com/ru-ru/library/system.string.trim(v=vs.110).aspx) | Удаление пробелов или знаков, указанных в массиве знаков, из начала и конца строки. |
| [String.TrimEnd](https://msdn.microsoft.com/ru-ru/library/system.string.trimend(v=vs.110).aspx) | Удаление знаков, указанных в массиве знаков, из конца строки. |
| [String.TrimStart](https://msdn.microsoft.com/ru-ru/library/system.string.trimstart(v=vs.110).aspx) | Удаление знаков, указанных в массиве знаков, из начала строки. |
| [String.Remove](https://msdn.microsoft.com/ru-ru/library/system.string.remove(v=vs.110).aspx) | Удаление указанного числа знаков из указанной позиции индекса в строке. |

## **Trim**

Простым методом удаления пробелов из обоих концов строки является метод [String.Trim](https://msdn.microsoft.com/ru-ru/library/t97s7bs3(v=vs.110).aspx), использование которого показано в следующем примере.

string MyString = " Big ";

Console.WriteLine("Hello{0}World!", MyString);

string TrimString = MyString.Trim();

Console.WriteLine("Hello{0}World!", TrimString);

// The example displays the following output:

// Hello Big World!

// HelloBigWorld!

Кроме того, можно удалить символы, указанные в массиве знаков из начала и конца строки. Следующий пример удаляет символы пробелов, точки и звездочки.

using System;

public class Example

{

public static void Main()

{

String header = "\* A Short String. \*";

Console.WriteLine(header);

Console.WriteLine(header.Trim( new Char[] { ' ', '\*', '.' } ));

}

}

// The example displays the following output:

// \* A Short String. \*

// A Short String

## **TrimEnd**

Метод String.TrimEnd удаляет знаки из конца строки, создавая новый объект строки. Массив знаков передается этому методу для указания удаляемых знаков. Порядок элементов в массиве знаков не влияет на операцию сокращения. Сокращение прекращается, если найденный знак не указан в массиве.

В следующем примере удаляются последние буквы строки с помощью метода TrimEnd. В этом примере положение знаков 'r' и 'W' изменено для иллюстрации того, что порядок знаков в массиве не имеет значения. Обратите внимание, что этот код удаляет последнее слово MyString и часть первого.

string MyString = "Hello World!";

char[] MyChar = {'r','o','W','l','d','!',' '};

string NewString = MyString.TrimEnd(MyChar);

Console.WriteLine(NewString);

Этот код выводит текст He на консоль.

В следующем примере с помощью метода TrimEnd удаляется последнее слово строки. В этом коде за словом Hello следует запятая, и, поскольку запятая не указана в массиве знаков для сокращения, сокращение прекращается на запятой.

string MyString = "Hello, World!";

char[] MyChar = {'r','o','W','l','d','!',' '};

string NewString = MyString.TrimEnd(MyChar);

Console.WriteLine(NewString);

Этот код выводит текст Hello, на консоль.

## **TrimStart**

Метод String.TrimStart аналогичен методу String.TrimEnd, за исключением того, что он создает новую строку, удаляя знаки из начала существующего объекта строки. Массив знаков передается методу TrimStart для указания удаляемых знаков. Как и в методе TrimEnd, порядок элементов в массиве знаков не влияет на операцию сокращения. Сокращение прекращается, если найденный знак не указан в массиве.

В следующем примере удаляется первое слово строки. В этом примере положение знаков 'l' и 'H' изменено для иллюстрации того, что порядок знаков в массиве не имеет значения.

string MyString = "Hello World!";

char[] MyChar = {'e', 'H','l','o',' ' };

string NewString = MyString.TrimStart(MyChar);

Console.WriteLine(NewString);

Этот код выводит текст World! на консоль.

## **Remove**

Метод [String.Remove](https://msdn.microsoft.com/ru-ru/library/9ad138yc(v=vs.110).aspx) удаляет указанное число знаков, начиная с указанного места в существующей строке. В этом методе предполагается, что индексация начинается с нуля.

В следующем примере удаляются десять знаков из строки, начиная с пятой позиции индекса строки, начинающегося с нуля.

string MyString = "Hello Beautiful World!";

Console.WriteLine(MyString.Remove(5,10));

// The example displays the following output:

// Hello World!

Можно также удалить из строки указанный символ или подстроку, вызвав метод [String.Replace(String, String)](https://msdn.microsoft.com/ru-ru/library/fk49wtc1(v=vs.110).aspx) с указанием пустой строки ([String.Empty](https://msdn.microsoft.com/ru-ru/library/system.string.empty(v=vs.110).aspx)) в качестве замены. Следующий пример удаляет все запятые из строки.

using System;

public class Example

{

public static void Main()

{

String phrase = "a cold, dark night";

Console.WriteLine("Before: {0}", phrase);

phrase = phrase.Replace(",", "");

Console.WriteLine("After: {0}", phrase);

}

}

// The example displays the following output:

// Before: a cold, dark night

// After: a cold dark night

## **Заполнение строк в .NET Framework**

Для создания новой строки, состоящей из исходной строки, дополненной знаками до указанной общей длины с начала или с конца, следует использовать один из следующих методов класса [String](https://msdn.microsoft.com/ru-ru/library/system.string(v=vs.110).aspx). В качестве заполняющего знака, который, соответственно, будет повторяться либо справа, либо слева, может использоваться пробел или знак, заданный в явной форме.

|  |  |
| --- | --- |
| **Название метода** | **Применение** |
| [String.PadLeft](https://msdn.microsoft.com/ru-ru/library/system.string.padleft(v=vs.110).aspx) | Дополняет строку до указанной общей длины знаками с начала. |
| [String.PadRight](https://msdn.microsoft.com/ru-ru/library/system.string.padright(v=vs.110).aspx) | Дополняет строку до указанной общей длины знаками с конца. |

**PadLeft**

Метод [String.PadLeft](https://msdn.microsoft.com/ru-ru/library/system.string.padleft(v=vs.110).aspx) создает новую строку, присоединяя к исходной строке знаки с начала в количестве, необходимом для достижения указанной общей длины. В методе [String.PadLeft(Int32)](https://msdn.microsoft.com/ru-ru/library/0zk6ydzx(v=vs.110).aspx) в качестве заполняющих знаков используются пробелы, а метод [String.PadLeft(Int32, Char)](https://msdn.microsoft.com/ru-ru/library/92h5dc07(v=vs.110).aspx) позволяет задать заполняющий знак в явной форме.

В следующем примере кода метод [PadLeft](https://msdn.microsoft.com/ru-ru/library/system.string.padleft(v=vs.110).aspx) используется для создания новой строки длиной в двадцать знаков. Этот пример выводит на консоль текст "--------Hello World!".

string MyString = "Hello World!";

Console.WriteLine(MyString.PadLeft(20, '-'));

**PadRight**

Метод [String.PadRight](https://msdn.microsoft.com/ru-ru/library/system.string.padright(v=vs.110).aspx) создает новую строку, присоединяя к исходной строке знаки с конца в количестве, необходимом для достижения указанной общей длины. В методе [String.PadRight(Int32)](https://msdn.microsoft.com/ru-ru/library/34d75d7s(v=vs.110).aspx) в качестве заполняющих знаков используются пробелы, а метод [String.PadRight(Int32, Char)](https://msdn.microsoft.com/ru-ru/library/36f2hz3a(v=vs.110).aspx) позволяет задать заполняющий знак в явной форме.

В следующем примере кода метод [PadRight](https://msdn.microsoft.com/ru-ru/library/system.string.padright(v=vs.110).aspx) используется для создания новой строки длиной в двадцать знаков. Этот пример выводит на консоль текст "Hello World!--------".

string MyString = "Hello World!";

Console.WriteLine(MyString.PadRight(20, '-'));

**Сравнение строк в .NET Framework**

Платформа .NET Framework обеспечивает несколько методов для сравнения значений строк. В таблице ниже перечислены и описаны методы сравнения значений.

| **Имя метода** | **Применение** |
| --- | --- |
| [String.Compare](https://msdn.microsoft.com/ru-ru/library/system.string.compare(v=vs.110).aspx) | Сравнивает значения двух строк. Возвращает целочисленное значение. |
| [String.CompareOrdinal](https://msdn.microsoft.com/ru-ru/library/system.string.compareordinal(v=vs.110).aspx) | Сравнивает две строки без учета локального языка и региональных параметров. Возвращает целочисленное значение. |
| [String.CompareTo](https://msdn.microsoft.com/ru-ru/library/system.string.compareto(v=vs.110).aspx) | Сравнивает текущий строковый объект с другой строкой. Возвращает целочисленное значение. |
| [String.StartsWith](https://msdn.microsoft.com/ru-ru/library/system.string.startswith(v=vs.110).aspx) | Определяет, начинается ли строка с переданной строки. Возвращает логическое значение. |
| [String.EndsWith](https://msdn.microsoft.com/ru-ru/library/system.string.endswith(v=vs.110).aspx) | Определяет, заканчивается ли строка переданной строкой. Возвращает логическое значение. |
| [String.Equals](https://msdn.microsoft.com/ru-ru/library/system.string.equals(v=vs.110).aspx) | Определяет, совпадают ли две строки. Возвращает логическое значение. |
| [String.IndexOf](https://msdn.microsoft.com/ru-ru/library/system.string.indexof(v=vs.110).aspx) | Возвращает индекс позиции символа или строки начиная с начала проверяемой строки. Возвращает целочисленное значение. |
| [String.LastIndexOf](https://msdn.microsoft.com/ru-ru/library/system.string.lastindexof(v=vs.110).aspx) | Возвращает индекс позиции символа или строки начиная с конца проверяемой строки. Возвращает целочисленное значение. |

**Сравнение**

Статический метод [String.Compare](https://msdn.microsoft.com/ru-ru/library/7aaf32ef(v=vs.110).aspx) позволяет тщательно сравнивать две строки. Этот метод учитывает язык и региональные параметры. Эту функцию можно использовать для сравнения двух строк или подстрок двух строк. Кроме того, имеются перегруженные методы, которые учитывают или не учитывают регистр и вариативность языка и региональных параметров. В таблице ниже приведены три целочисленных значения, которые может возвращать этот метод.

|  |  |
| --- | --- |
| **Возвращаемое значение** | **Условие** |
| Отрицательное целое число | Первая строка предшествует второй в порядке сортировки.  -или-  Первая строка имеет значение **null**. |
| 0 | Первая и вторая строка равны.  -или-  Обе строки имеют значение **null**. |
| Положительное целое число  -или-  1 | Первая строка следует за второй в порядке сортировки.  -или-  Вторая строка имеет значение **null**. |

В примере ниже метод [String.Compare](https://msdn.microsoft.com/ru-ru/library/84787k22(v=vs.110).aspx) используется для определения относительных значений двух строк.

string string1 = "Hello World!";

Console.WriteLine(String.Compare(string1, "hello world?"));

Этот пример выводит на консоль значение -1.

В предыдущем примере по умолчанию учитывается язык и региональные параметры. Для сравнения строк без учета языка и региональных параметров используйте перегрузку метода [String.Compare](https://msdn.microsoft.com/ru-ru/library/84787k22(v=vs.110).aspx), которая позволяет указать язык и региональные параметры с помощью параметра *culture*.

**CompareOrdinal**

Возвращаемые этим методом значения идентичны значениям, возвращаемым методом Compare в предыдущей таблице.

В примере ниже метод CompareOrdinal используется для сравнения значений двух строк.

string string1 = "Hello World!";

Console.WriteLine(String.CompareOrdinal(string1, "hello world!"));

Этот пример выводит на консоль значение -32.

**CompareTo**

Возвращаемые этим методом значения идентичны значениям, возвращаемым методом [String.Compare](https://msdn.microsoft.com/ru-ru/library/84787k22(v=vs.110).aspx) в предыдущей таблице. В примере ниже метод [String.CompareTo](https://msdn.microsoft.com/ru-ru/library/35f0x18w(v=vs.110).aspx) используется для сравнения объекта string1 с объектом string2.

string string1 = "Hello World";

string string2 = "Hello World!";

int MyInt = string1.CompareTo(string2);

Console.WriteLine( MyInt );

Этот пример выводит на консоль значение -1.

Все перегрузки метода [String.CompareTo](https://msdn.microsoft.com/ru-ru/library/35f0x18w(v=vs.110).aspx) по умолчанию выполняют сравнение с учетом языка и региональных параметров и регистра. У этого метода нет перегрузок, позволяющих выполнять сравнение без учета языка и региональных параметров. В целях повышения ясности кода рекомендуется использовать вместо него метод **String.Compare**, указывая [CultureInfo.CurrentCulture](https://msdn.microsoft.com/ru-ru/library/system.globalization.cultureinfo.currentculture(v=vs.110).aspx) для операций с учетом языка и региональных параметров и [CultureInfo.InvariantCulture](https://msdn.microsoft.com/ru-ru/library/system.globalization.cultureinfo.invariantculture(v=vs.110).aspx) для операций без учета языка и региональных параметров.

**Equals**

Этот метод учитывает регистр и возвращает логическое значение trueили false. Метод можно вызывать из существующего класса, как показано в следующем примере. В примере ниже метод Equalsиспользуется для определения того, содержит ли строковый объект фразу "Hello World".

string string1 = "Hello World";

Console.WriteLine(string1.Equals("Hello World"));

Этот пример выводит на консоль значение True.

Этот метод также можно использовать как статический. В примере ниже два строковых объекта сравниваются с помощью статического метода.

string string1 = "Hello World";

string string2 = "Hello World";

Console.WriteLine(String.Equals(string1, string2));

Этот пример выводит на консоль значение True.

**StartsWith и EndsWith**

Метод String**.**StartsWith можно использовать для определения того, начинается ли строковый объект с тех же символов, которые включает другая строка. Этот метод учитывает регистр и возвращает значение true, если текущий строковый объект начинается с переданной строки, и значение false в противном случае. В примере ниже этот метод используется для определения того, начинается ли строковый объект со слова "Hello".

string string1 = "Hello World";

Console.WriteLine(string1.StartsWith("Hello"));

Этот пример выводит на консоль значение True.

Метод String**.**EndsWith также возвращает логическое значение. В примере ниже конец строки проверяется с помощью метода **EndsWith**.

string string1 = "Hello World";

Console.WriteLine(string1.EndsWith("Hello"));

Этот пример выводит на консоль значение False.

**IndexOf и LastIndexOf**

С помощью метода String.IndexOf можно определить позицию первого вхождения конкретного символа в строку. Этот метод учитывает регистр и начинает отсчет с начала строки. Он возвращает позицию переданного символа, используя отсчитываемый от нуля индекс. Если символ не удается найти, возвращается значение –1.

В примере ниже метод IndexOf используется для поиска первого вхождения символа "l" в строку.

string string1 = "Hello World";

Console.WriteLine(string1.IndexOf('l'));

Этот пример выводит на консоль значение 2.

Данный метод учитывает регистр и использует отсчитываемый от нуля индекс.

В примере ниже метод LastIndexOf используется для поиска последнего вхождения символа "l" в строку.

string string1 = "Hello World";

Console.WriteLine(string1.LastIndexOf('l'));

Этот пример выводит на консоль значение 9.

Оба метода полезно использовать в сочетании с методом String.Remove. Для получения позиции символа используется метод IndexOf или **LastIndexOf**, после чего эта позиция передается методу Remove для удаления символа или начинающегося с него слова.

**Смена регистра в .NET Framework**

При написании приложения, которое принимает входные данные от пользователя, невозможно предугадать, какой регистр будет использоваться для ввода данных. Часто требуется обеспечить согласованность регистра строк, особенно если они отображаются в пользовательском интерфейсе. В таблице ниже описаны три метода изменения регистра. Для первых двух методов имеются перегруженные варианты, учитывающие язык и региональные параметры.

|  |  |
| --- | --- |
| **Имя метода** | **Применение** |
| String.ToUpper | Преобразует все символы в строке в верхний регистр. |
| String.ToLower | Преобразует все символы в строке в нижний регистр. |
| TextInfo.ToTitleCase | Преобразует первые символы всех слов в строке в заглавные. |

**Примечание:** Обратите внимание, что методы [String.ToUppe](https://msdn.microsoft.com/ru-ru/library/system.string.toupper(v=vs.110).aspx)r и [String.ToLower](https://msdn.microsoft.com/ru-ru/library/system.string.tolower(v=vs.110).aspx) не следует использовать для преобразования строк с целью их сравнения или проверки на равенство. Подробнее см. ниже.

**Сравнение строк, содержащих символы в разных регистрах**

Чтобы сравнить строки, содержащие символы в разных регистрах, для их упорядочения, вызовите одну из перегрузок метода [String.CompareTo](https://msdn.microsoft.com/ru-ru/library/35f0x18w(v=vs.110).aspx) с параметром *comparisonType* и укажите одно из значениий

[StringComparison.CurrentCultureIgnoreCase](https://msdn.microsoft.com/ru-ru/library/system.stringcomparison(v=vs.110).aspx), [StringComparison.OrdinalIgnoreCase](https://msdn.microsoft.com/ru-ru/library/system.stringcomparison(v=vs.110).aspx) или

[StringComparison.InvariantCultureIgnoreCase](https://msdn.microsoft.com/ru-ru/library/system.stringcomparison(v=vs.110).aspx) для аргумента *comparisonType*.

Для сравнения с использованием определенного языка, отличного от текущего языка и региональных параметров, вызовите перегрузку метода [String.CompareTo](https://msdn.microsoft.com/ru-ru/library/35f0x18w(v=vs.110).aspx) с параметрами *culture* и *options* и укажите значение [CompareOptions.IgnoreCase](https://msdn.microsoft.com/ru-ru/library/system.globalization.compareoptions(v=vs.110).aspx) для аргумента *options*.

Чтобы сравнить строки, содержащие символы в разных регистрах, для определения их равенства, вызовите одну из перегрузок метода [String.Equals](https://msdn.microsoft.com/ru-ru/library/858x0yyx(v=vs.110).aspx) с параметром *comparisonType* и укажите значение

[StringComparison.OrdinalIgnoreCaseStringComparison, CurrentCultureIgnoreCase](https://msdn.microsoft.com/ru-ru/library/system.stringcomparison(v=vs.110).aspx) или [StringComparison.InvariantCultureIgnoreCase](https://msdn.microsoft.com/ru-ru/library/system.stringcomparison(v=vs.110).aspx) для аргумента *comparisonType*.

**ToUpper**

Метод [String.ToUpper](https://msdn.microsoft.com/ru-ru/library/ewdd6aed(v=vs.110).aspx) преобразует все символы в строке в верхний регистр. В примере ниже смешанный регистр строки "Hello World!" изменяется на верхний.

string properString = "Hello World!";

Console.WriteLine(properString.ToUpper());

// This example displays the following output:

// HELLO WORLD!

В предыдущем примере по умолчанию учитываются язык и региональные параметры. В нем применяются соглашения об использовании регистров, действующие для текущих языка и региональных параметров. Чтобы изменить регистр с учетом языка и региональных параметров или применить соглашения об использовании регистров, принятые для определенного языка и региональных параметров, воспользуйтесь перегруженным методом [String.ToUpper(CultureInfo)](https://msdn.microsoft.com/ru-ru/library/24kc78ka(v=vs.110).aspx) и укажите в качестве параметра *culture* значение [CultureInfo.InvariantCulture](https://msdn.microsoft.com/ru-ru/library/system.globalization.cultureinfo.invariantculture(v=vs.110).aspx) или [System.Globalization.CultureInfo](https://msdn.microsoft.com/ru-ru/library/system.globalization.cultureinfo(v=vs.110).aspx), определяющее язык и региональные параметры.

**ToLower**

Метод [String.ToLower](https://msdn.microsoft.com/ru-ru/library/e78f86at(v=vs.110).aspx) аналогичен предыдущему методу, однако преобразует все символы в строке в нижний регистр. В примере ниже регистр строки "Hello World!" изменяется на нижний.

string properString = "Hello World!";

Console.WriteLine(properString.ToLower());

// This example displays the following output:

// hello world!

**ToTitleCase**

Метод [TextInfo.ToTitleCase](https://msdn.microsoft.com/ru-ru/library/system.globalization.textinfo.totitlecase(v=vs.110).aspx) преобразует первый символ каждого слова в верхний регистр, а остальные символы – в нижний. Однако слова, состоящие только из прописных букв, считаются сокращениями и не преобразуются.

Метод [TextInfo.ToTitleCase](https://msdn.microsoft.com/ru-ru/library/system.globalization.textinfo.totitlecase(v=vs.110).aspx) учитывает регистр, то есть он использует соглашения об использовании регистров, действующие для определенного языка и региональных параметров. Чтобы вызвать этот метод, сначала нужно получить объект [TextInfo](https://msdn.microsoft.com/ru-ru/library/system.globalization.textinfo(v=vs.110).aspx), представляющий соглашения об использовании регистров, из свойства [CultureInfo.TextInfo](https://msdn.microsoft.com/ru-ru/library/system.globalization.cultureinfo.textinfo(v=vs.110).aspx) конкретного языка и региональных параметров.

В примере ниже каждая строка из массива передается в метод [TextInfo.ToTitleCase](https://msdn.microsoft.com/ru-ru/library/system.globalization.textinfo.totitlecase(v=vs.110).aspx). Среди строк есть как строки заголовков, так и сокращения. Строки преобразуются в последовательности слов, начинающихся с заглавных букв, согласно соглашениям об использовании регистров для языка и региональных параметров "Английский (США)".

using System;

using System.Globalization;

public class Example

{

public static void Main()

{

string[] values = { "a tale of two cities", "gROWL to the rescue",

"inside the US government", "sports and MLB baseball",

"The Return of Sherlock Holmes", "UNICEF and children"};

TextInfo ti = CultureInfo.CurrentCulture.TextInfo;

foreach (var value in values)

Console.WriteLine("{0} --> {1}", value, ti.ToTitleCase(value));

}

}

// The example displays the following output:

// a tale of two cities --> A Tale Of Two Cities

// gROWL to the rescue --> Growl To The Rescue

// inside the US government --> Inside The US Government

// sports and MLB baseball --> Sports And MLB Baseball

// The Return of Sherlock Holmes --> The Return Of Sherlock Holmes

// UNICEF and children --> UNICEF And Children

Обратите внимание на то, что хотя метод [TextInfo.ToTitleCase](https://msdn.microsoft.com/ru-ru/library/system.globalization.textinfo.totitlecase(v=vs.110).aspx) и учитывает язык и региональные параметры, он не обеспечивает лингвистическую правильность использования прописных и строчных букв. Например, в предыдущем примере метод преобразует строку "a tale of two cities" в "A Tale Of Two Cities". При этом лингвистически правильным для языка и региональных параметров en-US будет преобразование "A Tale of Two Cities".

# Использование класса StringBuilder в .NET Framework

# Объект [String](https://msdn.microsoft.com/ru-ru/library/system.string(v=vs.110).aspx) является неизменяемым. Каждый раз при использовании одного из методов в классе [System.String](https://msdn.microsoft.com/ru-ru/library/system.string(v=vs.110).aspx) создается новый объект строки в памяти. При этом для нового объекта требуется новое выделение пространства. В тех случаях, когда необходимо выполнять повторяющиеся модификации строки, издержки, связанные с созданием нового объекта [String](https://msdn.microsoft.com/ru-ru/library/system.string(v=vs.110).aspx), могут оказаться значительными. Чтобы изменять строку без создания нового объекта, можно использовать класс [System.Text.StringBuilder](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder(v=vs.110).aspx). Например, использование класса [StringBuilder](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder(v=vs.110).aspx) может повысить производительность при соединении большого количества строк в цикле.

Класс [StringBuilder](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder(v=vs.110).aspx) находится в пространстве имен [System.Text](https://msdn.microsoft.com/ru-ru/library/system.text(v=vs.110).aspx).:

using System;

using System.Text;

## **Создание экземпляров объекта StringBuilder**

Можно создать новый экземпляр класса [StringBuilder](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder(v=vs.110).aspx) путем инициализации переменной с помощью одного из перегруженных методов конструктора, как показано в следующем примере.

StringBuilder MyStringBuilder = new StringBuilder("Hello World!");

## **Настройка емкости и длины**

Хотя [StringBuilder](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder(v=vs.110).aspx) является динамическим объектом, который позволяет вам развернуть ряд символов в строке, которые она инкапсулирует, вы можете указать максимальное число содержащихся в ней символов. Это значение называется емкостью объекта, и его не следует путать с длиной строки, которую содержит текущий [StringBuilder](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder(v=vs.110).aspx). Например, вы можете создать новый экземпляр класса [StringBuilder](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder(v=vs.110).aspx) со строкой Hello, длина которой составляет 5 символов, или указать, что максимальная емкость объекта — 25. При изменении [StringBuilder](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder(v=vs.110).aspx) размер не перераспределяется, пока не будет достигнута граница емкости. Когда это происходит, новое пространство выделяется автоматически, а емкость удваивается. Вы можете указать емкость класса [StringBuilder](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder(v=vs.110).aspx) с помощью одного из перегруженных конструкторов. В следующем примере показано, что объект MyStringBuilder можно развернуть максимум на 25 позиций.

StringBuilder MyStringBuilder = new StringBuilder("Hello World!", 25);

Кроме того, вы можете использовать свойство [Capacity](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder.capacity(v=vs.110).aspx) для чтения или записи, чтобы задать максимальную длину объекта. В следующем примере используется свойство **Capacity** для определения максимальной длины объекта.

MyStringBuilder.Capacity = 25;

## **Изменение строки StringBuilder**

В следующей таблице перечислены методы, которые можно использовать для изменения содержимого **StringBuilder**.

|  |  |
| --- | --- |
| **Имя метода** | **Применение** |
| [StringBuilder.Append](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder.append(v=vs.110).aspx) | Добавляет сведения в конец текущего **StringBuilder**. |
| [StringBuilder.AppendFormat](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder.appendformat(v=vs.110).aspx) | Заменяет отформатированным текстом описатель формата, переданный в строке. |
| [StringBuilder.Insert](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder.insert(v=vs.110).aspx) | Вставляет строку или объект в указанный индекс текущего **StringBuilder**. |
| [StringBuilder.Remove](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder.remove(v=vs.110).aspx) | Удаляет указанное количество символов из текущего **StringBuilder**. |
| [StringBuilder.Replace](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder.replace(v=vs.110).aspx) | Заменяет указанный символ в указанном индексе. |

### **Добавить**

Метод **Append** можно использовать для добавления текста или представления строки объекта к концу строки, представленной текущим**StringBuilder**. Следующий пример инициализирует **StringBuilder** с текстом Hello World, а затем добавляет текст в конец объекта.Пространство выделяется автоматически при необходимости.

StringBuilder MyStringBuilder = new StringBuilder("Hello World!");

MyStringBuilder.Append(" What a beautiful day.");

Console.WriteLine(MyStringBuilder);

// The example displays the following output:

// Hello World! What a beautiful day.

### **AppendFormat**

Метод [StringBuilder.AppendFormat](https://msdn.microsoft.com/ru-ru/library/hdekwk0b(v=vs.110).aspx) добавляет текст в конец объекта [StringBuilder](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder(v=vs.110).aspx). Он поддерживает функцию составного форматирования путем вызова реализации [IFormattable](https://msdn.microsoft.com/ru-ru/library/system.iformattable(v=vs.110).aspx) форматируемого объекта или объектов. Следовательно, он принимает строки стандартного формата для числовых значений, значений даты и времени, значений перечисления, строки пользовательского формата для чисел и дат и строки формата, определенные для пользовательских типов. Можно использовать этот метод для настройки формата переменных и добавления этих значений в [StringBuilder](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder(v=vs.110).aspx). В следующем примере используется метод [AppendFormat](https://msdn.microsoft.com/ru-ru/library/hdekwk0b(v=vs.110).aspx)для размещения целочисленного значения, отформатированного в качестве значения валюты в конце объекта [StringBuilder](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder(v=vs.110).aspx).

int MyInt = 25;

StringBuilder MyStringBuilder = new StringBuilder("Your total is ");

MyStringBuilder.AppendFormat("{0:C} ", MyInt);

Console.WriteLine(MyStringBuilder);

// The example displays the following output:

// Your total is $25.00

### **Insert**

Метод [Insert](https://msdn.microsoft.com/ru-ru/library/6tkz1aa7(v=vs.110).aspx) добавляет строку или объект в указанную позицию в текущем объекте [StringBuilder](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder(v=vs.110).aspx). В следующем примере этот метод используется для вставки слова в шестую позицию объекта [StringBuilder](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder(v=vs.110).aspx).

StringBuilder MyStringBuilder = new StringBuilder("Hello World!");

MyStringBuilder.Insert(6,"Beautiful ");

Console.WriteLine(MyStringBuilder);

// The example displays the following output:

// Hello Beautiful World!

### **Удалить**

Вы можете использовать метод **Remove**, чтобы удалить указанное количество символов из текущего объекта [StringBuilder](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder(v=vs.110).aspx), начиная с указанного индекса (с отсчетом с нуля). В следующем примере используется метод **Remove** для сокращения объекта [StringBuilder](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder(v=vs.110).aspx).

StringBuilder MyStringBuilder = new StringBuilder("Hello World!");

MyStringBuilder.Remove(5,7);

Console.WriteLine(MyStringBuilder);

// The example displays the following output:

// Hello

### **Заменить**

Метод **Replace** можно использовать для замены символов в объекте [StringBuilder](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder(v=vs.110).aspx) другими указанными символами. В следующем примере метод **Replace** используется для поиска объекта [StringBuilder](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder(v=vs.110).aspx) для всех экземпляров восклицательного знака (!) и замены их знаком вопроса (?).

StringBuilder MyStringBuilder = new StringBuilder("Hello World!");

MyStringBuilder.Replace('!', '?');

Console.WriteLine(MyStringBuilder);

// The example displays the following output:

// Hello World?

## **Преобразование объекта StringBuilder в строку**

Необходимо преобразовать объект [StringBuilder](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder(v=vs.110).aspx) в [String](https://msdn.microsoft.com/ru-ru/library/system.string(v=vs.110).aspx) перед тем, как вы сможете передать строку, представленную объектом [StringBuilder](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder(v=vs.110).aspx), методу, который содержит параметр [String](https://msdn.microsoft.com/ru-ru/library/system.string(v=vs.110).aspx), или отобразить ее в пользовательском интерфейсе. Это преобразование можно выполнить, вызвав метод [StringBuilder.ToString](https://msdn.microsoft.com/ru-ru/library/4b1063zt(v=vs.110).aspx). Следующий пример вызывает ряд методов [StringBuilder](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder(v=vs.110).aspx), а затем вызывает метод [StringBuilder.ToString()](https://msdn.microsoft.com/ru-ru/library/4b1063zt(v=vs.110).aspx) для отображения строки.

using System;

using System.Text;

public class Example

{

public static void Main()

{

StringBuilder sb = new StringBuilder();

bool flag = true;

string[] spellings = { "recieve", "receeve", "receive" };

sb.AppendFormat("Which of the following spellings is {0}:", flag);

sb.AppendLine();

for (int ctr = 0; ctr <= spellings.GetUpperBound(0); ctr++) {

sb.AppendFormat(" {0}. {1}", ctr, spellings[ctr]);

sb.AppendLine();

}

sb.AppendLine();

Console.WriteLine(sb.ToString());

}

}

// The example displays the following output:

// Which of the following spellings is True:

// 0. recieve

// 1. receeve

// 2. receive

ВАРИАНТЫ ИНДИВИДУАЛЬНЫХ ЗАДАНИЙ

Общая постановка задачи. Выполнить задание, рассматривая слова и строки как последовательность Char-символов, объекты типа System.String и типа System.Text.StringBuilder.

1. Даны два слова одинаковой длины. Подсчитать количество попарно одинаковых букв.
2. Даны два слова одинаковой длины. Подсчитать количество попарно различных букв.
3. Дана строка. Подсчитать количество содержащихся в ней цифр.
4. Дана строка. Подсчитать количество содержащихся в ней прописных букв.
5. Дана строка. Подсчитать количество содержащихся в ней строчных букв.
6. Даны 2 слова. Подсчитать сколько раз первая буква первого слова встречается во втором слове.
7. Дана строка, изображающая целое число. Найти сумму цифр этого числа.
8. Дана строка. Преобразовать все строчные буквы в прописные.
9. Дана строка. Преобразовать все прописные буквы в строчные.
10. Дана строка S и число N. Преобразовать строку S в строку длины N следующим образом: если длина строки S больше N, то отбросить первые символы, если длина строки S меньше N, то в ее начало добавить символы "." (точка).
11. Даны два числа: N1 и N2, и две строки: S1 и S2. Получить из этих строк новую строку, объединив N1 первых символов строки S1 и N2 последних символов строки S2.
12. Даны две строки: S1 и S2. Проверить, содержится ли строка S2 в строке S1. Если да, то вывести номер позиции, начиная с которой S2 содержится в S1, если нет, то вывести 0.
13. Даны две строки: S1 и S2. Определить количество вхождений строки S2 в строку S1.
14. Дана строка S и символ С. Удвоить каждое вхождение символа С в строку S.
15. Даны строки SI, S2 и символ С. Перед (после) каждого вхождения символа С в строку S1 вставить строку S2.
16. Дан текст. Вывести подстроку, расположенную между первой и второй точками исходного текста. Если в строке менее двух точек, то вывести весь исходный текст.
17. Дана строка, состоящая из слов, разделенных пробелами. Определить количество слов в строке.
18. Дана строка, состоящая из слов, разделенных пробелами. Определить длину самого короткого слова.
19. Дана строка, состоящая из слов, разделенных пробелами. Определить длину самого длинного слова.
20. Дана строка, состоящая из слов, разделенных пробелами. Вывести строку, содержащую эти же слова, но разделенные одним символом "." (точка). В конце точку не ставить.
21. Дана строка-предложение, содержащая избыточные пробелы. Преобразовать ее так, чтобы между словами был ровно один пробел.
22. Дана строка, содержащая полное имя файла, то есть имя диска, список каталогов (путь), собственно имя и расширение. Выделить из этой строки имя файла.
23. Дана строка, содержащая полное имя файла. Выделить из строки название последнего каталога (без символов "\"). Если файл содержится в корневом каталоге, то вывести символ "\".

ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ

1. Албахари, Дж. C#5.0. Справочник. Полное описание языка / Дж. Албахари, Б. Албахари. – 5-е изд. – М.: ООО «И.Д. Вильямс», 2014. – 1008 с.: ил.
2. Дейтел, П. Как программировать на Visual C# 2012 / П. Дейтел, Х. Дейтел. – 5-е изд. – СПб.: Питер, 2014. – 864 с.: ил.
3. Рихтер, Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# / Дж. Рихтер. – 4-е изд. – СПб.: Питер, 2013. – 896 с.: ил.
4. Троелсен, Э. Язык программирования C# 5.0 и платформа .NET 4.5 / Э. Троелсен. – 6-е изд. – М.: ООО «И.Д. Вильямс», 2013. – 1312 с.: ил.
5. Шилдт, Г. C#4.0: Полное руководство / Г. Шилдт. – М.: ООО «И.Д. Вильямс», 2011. – 1056 с.: ил.
6. Полное руководство по языку программирования С# 6.0 и платформе .NET 4.6 [Электронный ресурс] / Сайт о программировании. – metanit.com, 2012-2016. – Режим доступа: [http://metanit.com/sharp/ tutorial](http://metanit.com/sharp/%20tutorial). – Дата доступа: 20.08.2016.

# Структура Char [Электронный ресурс] / Microsoft. – Microsoft, 2016. – Режим доступа: https://msdn.microsoft.com/ru-ru/library/system.char(v=vs.110) .aspx. – Дата доступа: 15.08.2016.

Преподаватель Е.В. Багласова

|  |
| --- |
| Рассмотрено на заседании цикловой  комиссии программного обеспечения информационных технологий №10  Протокол №\_\_\_\_\_от «\_\_\_»\_\_\_\_2016  Председатель ЦК *( )* Т.Г.Багласова |