Строки

В C# строки являются объектами. Тип string относится к числу ссылочных.
Ключевое слово string является псевдонимом класса System.String, определенного в библиотеке классов для среды .NET Framework

Инициализация строк:

```
string s1= "Строки в C#";
string s2 = "First Line\nSecond Line";
string s3 = @"\\server\fileshare\helloworld.cs";
```

Чтобы создать повторяющуюся последовательность символов, можно использовать конструктор string:

```
Console.Write (new string ('*', 10));  // **********
```

Язык программирования C#

Строку можно также сконструировать из массива char. Метод ToCharArray вы-полняет обратное действие:

```
char[] charray = {'t', 'e', 's', 't'};
string s = new string (charray) ;  // s = “test"
charray = "Hello".ToCharArray();

Console.WriteLine(str);
Console.WriteLine(str[0]) ;  // ‘t’
```

С помощью индекса нельзя присвоить новое значение символу в строке

Класс типа string содержит ряд методов для обращения со строками.

Тип string реализует интерфейс IEmunerable<char>, т.е. по символам строки можно проходить с помощью foreach:

```
foreach (char c in "123") Console.Write (c + " "); // 1 2 3
```

Строки null и пустые строки

Пустая строка имеет нулевую длину. Чтобы создать пустую строку, можно применить либо литерал, либо статическое поле string.Empty.
Для проверки, пуста ли строка, можно либо выполнить сравнение эквивалентности, либо просмотреть свойство Length строки:

```
string empty = "";
Console.WriteLine (empty == "");              // True
Console.WriteLine (empty == string.Empty);  // True
Console.WriteLine (empty.Length == 0);        // True
```

Строки могут быть null:

```
string nullstring = null;
Console.WriteLine(nullstring == null);       // True
Console.WriteLine (nullstring == "") ;       // False
Console.WriteLine (nullstring.Length == 0)    // Генерируется исключение
                                              // NullReferenceException
```

Язык программирования C#

Для проверки строк на Null или Empty статические методы:
string.IsNullOrEmpty(), string.NotNullOrEmpty()

```
string nullstring = null;
if (String.IsNullOrEmpty(nullstring))
    Console.WriteLine("пустая или null-строка");
```

Язык программирования C#

string —ссылочный тип, однако, его операции эквивалентности следуют семантике типов значений

Проверка на равенство: оператор ==

Неравенство строк: !=

Тип string не поддерживает операции < и > для сравнений. Вместо них должен применяться метод CompareTo.

Конкатенация строк

Операция +

```
string s = "a" + "b";
```

Один из операндов может быть не строковым значением. В этом случае для него неявно будет вызван метод ToString.

```
string s = "a" + 5;   // a5
```

Поиск внутри строк

К простейшим методам поиска относятся: StartsWith, EndsWith и Contains. Все они возвращают true или false:

```
Console.WriteLine ("Строки в C#".EndsWith ("C#"));  // True
Console.WriteLine ("Строки в C#".Contains ("в C")); // True
```

Поиск, нечувствительный к регистру символов, с использованием правил, не зависящих от культуры:

```
Console.WriteLine ("abcdef".StartsWith ("aBc",
                   StringComparison.InvariantCultureIgnoreCase));
```

Метод IndexOf он возвращает позицию первого вхождения заданного символа или подстроки (или -1, если символ или подстрока не найдена):

```
Console.WriteLine ("abcde".IndexOf ("cd")); // 2
Console.WriteLine("abcde abcde".IndexOf("CD", 6,
                   StringComparison.CurrentCultureIgnoreCase)); // 8
```

Язык программирования C#

Метод LastIndexOf аналогичен IndexOf, но перемещается по строке в обратном направлении.

Метод IndexOfAny возвращает позицию первого вхождения любого символа из множества символов:

```
Console.WriteLine ("ab,cd ef".IndexOfAny (new char[] {'1',',','g'})); // 2
Console.WriteLine ("pas5wOrd".IndexOfAny("0123456789".ToCharArray() )); //3
```

Метод LastIndexOfAny аналогичен IndexOfAny, но в обратном направлении.

Методы для работы со строками

Поскольку класс String неизменяемый, все методы, которые “манипулируют” строкой, возвращают новую строку, оставляя исходную незатронутой (то же самое происходит при повторном присваивании строковой переменной).
Метод SubString извлекает подстроку:

```
Console.WriteLine ("12345".Substring (0, 3));      // 123
Console.WriteLine ("12345".Substring (1, 3));      // 234
Console.WriteLine ("12345".Substring (2));         // 345
```

Методы Insert и Remove вставляют либо удаляют символы в указанной позиции:

```
Console.WriteLine ("helloworld" .Insert (5, ",  "));     //  hello,  world
Console.WriteLine ("hello, world".Remove(5, 2));     //  helloworld
Console.WriteLine ("hello, world".Remove(5));    //  hello
```

Методы для работы со строками

Методы PadLeft и PadRight дополняют строку до заданной длины слева или справа заданным символом (или пробелом, если символ не указан):

```
Console.WriteLine ("12345".PadLeft (9, '*'));      // ****12345
Console.WriteLine ("12345".PadLeft (9));      //    12345
```

Методы TrimStart и TrimEnd удаляют указанные символы из начала или конца строки; метод Trim делает то и другое. По умолчанию эти функции удаляют пробельные символы (включающие пробелы, табуляции, символы новой строки и их вариации в Unicode):

```
Console.WriteLine (" abc \t\r\n ".Trim()); //abc
Console.WriteLine ("a iabc cvvav".Trim('a','v')); //    iabc c
```

Методы для работы со строками

Метод Replace заменяет все (неперекрывающиеся) вхождения заданного символа или подстроки:

```
Console.WriteLine ("to be done".Replace (" ", " | ") );  // to | be | done
Console.WriteLine ("to be done".Replace (" ", "")    );   // tobedone
```

Разделение и объединение строк

Метод Split разделяет строку на несколько:

```
string[] words = "Строки в C#".Split() ;
foreach (string word in words)
    Console.Write (word + " I "); // Строки I в I C#|
```

По умолчанию в качестве разделителей метод Split использует пробельные символы.

Методы для работы со строками

Использование split с массивом разделителей в качестве параметра:

```
Console.WriteLine ("Строки в C#".Split('р','и')) ;
foreach (string word in words)
      Console.Write (word + " I ");         //   Ст | ок |  в C# |
```

Статический метод Join выполняет противоположное действие.

```
string[] words = "Строки в C#".Split();
string together = string.Join (",", words); // Строки,в,C#
```

Статический метод Concat – объединение строк без учета разделителей. Метод Concat эквивалентен операции + (компилятор транслирует операцию + в вызов Concat):

```
string sentence = string.Concat ("Строки", "в", "C#");
string sameSentence = "Строки" + "в" + "C#";
```

Метод string.Format и смешанные форматные строки

Способ построения строк, которые содержат в себе переменные. Эти встроенные переменные (или значения) могут относиться к любому типу; метод Format вызывает для них ToString.

Главная строка, включающая встроенные переменные, называется смешанной форматной строкой.

При вызове методу String.Format предоставляется такая смешанная форматная строка, а за ней по очереди все встроенные переменные.

```
string composite = "It's {0} degrees in {1} on this {2} morning";
string s = string.Format(composite, 35, "Moscow",
                         DateTime.Now.DayOfWeek);
// s == "It's 35 degrees in Moscow on this Friday morning«
```

Метод string.Format и смешанные форматные строки

Начиная с версии C# 6, для получения тех же результатов можно применять интерполированные строковые литералы.

```
string s = $"It's hot this {DateTime.Now.DayOfWeek} morning";
```

Каждое число в фигурных скобках называется форматным элементом. Число соответствует позиции аргумента и за ним может дополнительно следовать:

- запятая и минимальная ширина (если значение отрицательное, тогда данные выравниваются влево, иначе — вправо.
- двоеточие и форматная строка.

```
string composite = "Name={0,-15} Credit Limit={1,15:C}";
Console.WriteLine (string.Format (composite, "Ivan", 500));
Console.WriteLine (string.Format (composite, "Ekaterina", 20000));
```

Вот как выглядит результат:

```
Name=Ivan            Credit Limit=         $500.00
Name=Ekaterina       Credit Limit=     $20,000.00
```

Эквивалентный код:

```
string s="Name=" + "Ekaterina".PadRight (15) +" CreditLimit=" + 500.ToString ("C").PadLeft (15);
```