Строки. Средства работы со строками

Строка в C# — это массив знаков, объявленный с помощью ключевого слова **string**. Строковый литерал объявляется с помощью кавычек, как показано в следующем примере.

```
string s = "Hello, World!";

// Объявление без инициализации.
string message1;

// Инициализация нулевой ссылкой - нет строки.
string message2 = null;

// Инициализация пустой строкой.
// Использование константы Empty вместо литерала "".
string message3 = System.String.Empty;

// Инициализация.
string oldPath = "c:\\Program Files\\Microsoft Visual Studio 8.0";

// Инициализация.
string newPath = @"c:\Program Files\Microsoft Visual Studio 9.0";


// Инициализация - const string.
const string message4 = "You can't get rid of me!";

// Массив и строка
char[] letters = { 'A', 'B', 'C' };
string alphabet = new string(letters);
```

Переприсваивание

Строки можно целиком переприсваивать:

```
string s1 = "Hello";
string s2 = s1; //переприсваиваем
```

Объединение строк

Можно объединять строки с помощью оператора +, как показано в следующем примере:

```
string s1 = "orange";
string s2 = "red";
s1 += s2;
Console.WriteLine(s1);  // напечатается "orangered"
```

Escape-знаки

Строки могут содержать escape-знаки, такие как "\n" (новая строка) и "\t" (табуляция). Пример:

```
string hello = "Hello\nWorld!";
Console.WriteLine(hello);
/* Напечатается
Hello
World!
*/
string s = "чтобы вставить \"кавычки\" в строку используем обратный слэш";
Console.WriteLine(s);
```

Если требуется добавить в строку обратную косую черту, перед ней нужно поставить еще одну обратную косую черту. Следующая строка:

```
string filePath = "\\\\My Documents\\";
Console.WriteLine(filePath);
/* Напечатается:
\\My Documents\
*/
```

Точные строки: символ @

Символ @ служит для того, чтобы конструктор строк пропускал escape-знаки и переносы строки. Следующие две строки являются идентичными:

```
string p1 = "\\\\My Documents\\My Files\\";
string p2 = @"\\My Documents\My Files\";
```

Чтобы поставить в точной строке знак двойных кавычек, нужно использовать по два таких знака, как показано в следующем примере:

```
string s = @"You say ""goodbye"" and I say ""hello""";
```

Доступ к отдельным знакам

Квадратные скобки [] служат для доступа к отдельным знакам в объекте **string**, но при этом возможен доступ только для чтения:

```
string str = "test";
char x = str[2];  // x = 's';

string s5 = "Printing backwards";
for (int i = 0; i < s5.Length; i++)
    Console.Write(s5[s5.Length - i - 1]);
   // напечатается "sdrawkcab gnitnirP"
```

Извлечение подстрок

Для извлечения подстроки из строки используется метод **Substring**.

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

Замена по образцу

Для замены подстроки в строке по образцу используется метод **Replace**.

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

Удаление фрагментов и вставка строк в строки

Может быть выполнена с помощью методов Remove и Insert:

```
string x = "ZX Spectrum";
x=x.Remove(2, 1);
x = x.Insert(2,"-");
Console.WriteLine(x);
```
Console.ReadKey();

Смена регистра

Чтобы изменить регистр букв в строке (сделать их заглавными или строчными) следует использовать **ToUpper()** или **ToLower()**, как показано в следующем примере:

```
string s6 = "Юфу";
Console.WriteLine(s6.ToUpper());
// Напечатается ЮФУ
Console.WriteLine(s6.ToLower());
// Напечатается юфу
Console.WriteLine(s6);
// Напечатается Юфу
```

Сравнения

Для строковых объектов существует метод CompareTo(), возвращающий целочисленное значение, зависящее от того, что одна строка может быть меньше (<), равна (==) или больше другой (>). При сравнении строк используется значение Юникода, при этом значение строчных букв меньше, чем значение заглавных.

```
string string1 = "ИИТ";
string string2 = "Иит";

int result = string1.CompareTo(string2);
if (result > 0)
     Console.WriteLine("{0} больше чем {1}", string1, string2);
  else if (result == 0)
     Console.WriteLine("{0} равно {1}", string1, string2);
  else if (result < 0)
      Console.WriteLine("{0} меньше чем {1}", string1, string2);
 // Напечатается ИИТ больше чем Иит
```

Разбиение строк

Следующий пример кода демонстрирует возможность разбора строки при помощи метода String.Split. Работа метода заключается в возврате массива строк, в котором каждый элемент представляет слово. В качестве ввода **Split** принимает массив символов, определяющих какие символы должны использоваться в качестве разделителей. В этом примере используются пробелы, запятые, точки, двоеточия и табуляция. Массив, содержащий эти разделители, передается в **Split**, и каждое слово в предложении выводится отдельно при помощи результирующего массива строк.

Пример: разбить предложение на слова

```
char razdelitel = ' ';
string text = "Шла Саша по шоссе и сосала сушку";
Console.WriteLine("Исходный текст: '{0}'", text);
string[] words = text.Split(razdelitel);
Console.WriteLine("{0} слов в тексте:", words.Length);
foreach (string s in words)
    Console.WriteLine(s);
```

В качестве разделителя может выступать массив символов.

```
char[] delimiterChars = { ' ', ',', '.', ':', '\t' };

string text = "one\ttwo three:four,five six seven";
Console.WriteLine("Original text: '{0}'", text);
string[] words = text.Split(delimiterChars);
Console.WriteLine("{0} words in text:", words.Length);
foreach (string s in words)
        Console.WriteLine(s);
```

Пример: Ввести текстовую строку. Напечатать слова, в которых первая буква встречается еще хотя бы раз.

```
Console.Write("Введите предложение: ");
string s=Console.ReadLine();
string[] words= s.Split(' ','.',',');
foreach (string word in words)
  {
     char c=word[0];
     bool flag = false;
     int i=1;
     while (i < word.Length & !flag)
       {
         if (word[i] == c) flag = true;
         i++;
       }
     if (flag) Console.WriteLine(word);
  }
Console.ReadKey();
```

Таблица 7. Динамические методы класса String

Метод	**Описание**
Insert	Вставляет подстроку в заданную позицию
Remove	Удаляет подстроку в заданной позиции
Replace	Заменяет подстроку в заданной позиции на новую подстроку
Substring	Выделяет подстроку в заданной позиции
IndexOf, IndexOfAny, LastIndexOf, LastIndexOfAny	Определяются индексы первого и последнего вхождения заданной подстроки или любого символа из заданного набора
StartsWith, EndsWith	Возвращается true или false, в зависимости от того, начинается или заканчивается строка заданной подстрокой
PadLeft, PadRight	Выполняет набивку нужным числом пробелов в начале и в конце строки
Trim,TrimStart, TrimEnd	Обратные операции к методам Pad. Удаляются пробелы в начале и в конце строки, или только с одного ее конца
ToCharArray	Преобразование строки в массив символов