Основы программирования

Самойлов Михаил Юрьевич

Строки

Большое количество задач, которые могут встретиться при разработке приложений, так или иначе связано с обработкой строк – разбор веб-страниц, поиск в тексте, какие-то аналитические задачи, связанные с извлечением нужной информации из текста и т.д.

Поэтому работе со строками уделяется особое внимание.

Строки

В языке C# строковые значения представляет тип string, а вся функциональность работы с данным типом сосредоточена в классе System.String. Собственно string является псевдонимом для класса System.String. Объекты этого класса представляют текст как последовательность символов Unicode. Максимальный размер объекта String может составлять в памяти 2 ГБ, или около 1 миллиарда символов.

Создание строк

Создавать сроки можно, как используя переменную типа string и присваивая ей значение, так и применяя один из конструкторов класса String:

```
string s1 = "hello";
string s2 = new String('a', 6); // результатом будет строка "aaaaaa"
string s3 = new String(new char[]{'w', 'o', 'r', 'l', 'd'});
```

Строка как набор символов

Так как строка хранит коллекцию символов, в ней определен индексатор для доступа к этим символам. Применяя индексатор, мы можем обратиться к строке как к массиву символов и получить по индексу любой из ее символов.

```
string s1 = "hello";
char ch1 = s1[1]; // символ 'e'
Console.WriteLine(ch1);
Console.WriteLine(s1.Length);
```

Используя свойство Length, как и в обычном массиве, можно получить длину строки.

Основные методы строк

- Compare: сравнивает две строки с учетом текущей культуры (локали) пользователя
- CompareOrdinal: сравнивает две строки без учета локали
- Contains: определяет, содержится ли подстрока в строке
- Concat: соединяет строки
- CopyTo: копирует часть строки или всю строку в другую строку

Основные методы строк

- EndsWith: определяет, совпадает ли конец строки с подстрокой
- Format: форматирует строку
- IndexOf: находит индекс первого вхождения символа или подстроки в строке
- Insert: вставляет в строку подстроку
- Join: соединяет элементы массива строк
- LastIndexOf: находит индекс последнего вхождения символа или подстроки в строке

Основные методы строк

- Replace: замещает в строке символ или подстроку другим символом или подстрокой
- Split: разделяет одну строку на массив строк
- Substring: извлекает из строки подстроку, начиная с указанной позиции
- ToLower: переводит все символы строки в нижний регистр
- ToUpper: переводит все символы строки в верхний регистр
- Trim: удаляет начальные и конечные пробелы из строки

Конкатенация

Конкатенация строк или объединение может производиться как с помощью операции +, так и с помощью метода Concat:

```
string s1 = "hello";
string s2 = "world";
string s3 = s1 + " " + s2; // результат: строка "hello world"
string s4 = String.Concat(s3, "!!!"); // результат: строка "hello world!!!"
Console.WriteLine(s4);
```

Конкатенация

Метод Concat является статическим методом класса String, принимающим в качестве параметров две строки. Также имеются другие версии метода, принимающие другое количество параметров.

Конкатенация

Для объединения строк также может использоваться метод Join:

```
string s5 = "apple";
string s6 = "a day";
string s7 = "keeps";
string s8 = "a doctor";
string s9 = "away";
string[] values = new string[] { s5, s6, s7, s8, s9 };

String s10 = String.Join(" ", values);
// результат: строка "apple a day keeps a doctor away"
```

Конкатенация

Метод Join также является статическим. Использованная выше версия метода получает два параметра: строку-разделитель (в данном случае пробел) и массив строк, которые будут соединяться и разделяться разделителем.

Сравнение строк

Для сравнения строк применяется статический метод Compare:

```
string s1 = "hello";
string s2 = "world";
int result = String.Compare(s1, s2);
if (result<0)
    Console.WriteLine("Строка s1 перед строкой s2");
else if (result > 0)
    Console.WriteLine("Строка s1 стоит после строки s2");
else
    Console.WriteLine("Строки s1 и s2 идентичны");
// результатом будет "Строка s1 перед строкой s2"
```

Сравнение строк

Данная версия метода Compare принимает две строки и возвращает число. Если первая строка по алфавиту стоит выше второй, то возвращается число меньше нуля. В противном случае возвращается число больше нуля. И третий случай - если строки равны, то возвращается число 0.

В данном случае так как символ h по алфавиту стоит выше символа w, то и первая строка будет стоять выше.

Поиск в строке

С помощью метода IndexOf мы можем определить индекс первого вхождения отдельного символа или подстроки в строке:

```
string s1 = "hello world";
char ch = 'o';
int indexOfChar = s1.IndexOf(ch); // равно 4
Console.WriteLine(indexOfChar);

string subString = "wor";
int indexOfSubstring = s1.IndexOf(subString); // равно 6
Console.WriteLine(indexOfSubstring);
```

Поиск в строке

Подобным образом действует метод LastIndexOf, только находит индекс последнего вхождения символа или подстроки в строку.

Еще одна группа методов позволяет узнать начинается или заканчивается ли строка на определенную подстроку. Для этого предназначены методы StartsWith и EndsWith.

Разделение строк

С помощью функции Split мы можем разделить строку на массив подстрок. В качестве параметра функция Split принимает массив символов или строк, которые и будут служить разделителями. Например, подсчитаем количество слов в сроке, разделив ее по пробельным символам:

```
string text = "Мама мыла раму с мылом";
string[] words = text.Split(new char[] { ' ' });
foreach (string s in words)
{
    Console.WriteLine(s);
}
```

Обрезка строки

Для обрезки начальных или концевых символов используется функция Trim:

```
string text = " hello world ";

text = text.Trim(); // результат "hello world"
text = text.Trim(new char[] { 'd', 'h' }); // результат "ello worl"
```

Обрезка строки

Функция Trim без параметров обрезает начальные и конечные пробелы и возвращает обрезанную строку. Чтобы явным образом указать, какие начальные и конечные символы следует обрезать, мы можем передать в функцию массив этих символов.

Эта функция имеет частичные аналоги: функция TrimStart обрезает начальные символы, а функция TrimEnd обрезает конечные символы.

Обрезка строки

Обрезать определенную часть строки позволяет функция Substring:

```
string text = "Хороший день";
// обрезаем начиная с третьего символа
text = text.Substring(2);
// результат "роший день"
Console.WriteLine(text);
// обрезаем сначала до последних двух символов
text = text.Substring(0, text.Length - 2);
// результат "роший де"
 Console.WriteLine(text);
```

Обрезка строки

Функция Substring также возвращает обрезанную строку. В качестве параметра первая использованная версия применяет индекс, начиная с которого надо обрезать строку. Вторая версия применяет два параметра - индекс начала обрезки и длину вырезаемой части строки.

Вставка

Для вставки одной строки в другую применяется функция Insert:

```
string text = "Хороший день";
string subString = "замечательный ";

text = text.Insert(8, subString);
Console.WriteLine(text);
```

Первым параметром в функции Insert является индекс, по которому надо вставлять подстроку, а второй параметр - собственно подстрока.

Удаление строк

Удалить часть строки помогает метод Remove:

```
string text = "Хороший день";
// индекс последнего символа
int ind = text.Length - 1;
// вырезаем последний символ
text = text.Remove(ind);
Console.WriteLine(text);

// вырезаем первые два символа
text = text.Remove(0, 2);
```

Удаление строк

Первая версия метода Remove принимает индекс в строке, начиная с которого надо удалить все символы. Вторая версия принимает еще один параметр - сколько символов надо удалить.

Замена

Чтобы заменить один символ или подстроку на другую, применяется метод Replace:

```
string text = "хороший день";

text = text.Replace("хороший", "плохой");
Console.WriteLine(text);

text = text.Replace("о", "");
Console.WriteLine(text);
```

Замена

Во втором случае применения функции Replace строка из одного символа "о" заменяется на пустую строку, то есть фактически удаляется из текста. Подобным способом легко удалять какой-то определенный текст в строках

Смена регистра

Для приведения строки к верхнему и нижнему регистру используются соответственно функции ToUpper() и ToLower():

```
string hello = "Hello world!";

Console.WriteLine(hello.ToLower()); // hello world!
Console.WriteLine(hello.ToUpper()); // HELLO WORLD!
```