Классы string и stringBilder. Регулярные выражения

Лекция №10

Объекты класса **String**

У класса **String** достаточно много конструкторов. Они позволяют сконструировать строку из:

- - символа, повторенного заданное число раз;
- - массива символов **char[ ]**;
- - части массива символов.

- **public void TestDeclStrings()**

```
{
string world = "Мир";
//string s1 = new string("s1");
//string s2 = new string();
string sssss = new string('s',5);
char[] yes = "Yes".ToCharArray();
string stryes = new string(yes);
string strye = new string(yes,0,2);
Console.WriteLine("world = {0}; sssss={1};
stryes={2};"+
" strye= {3}", world, sssss, stryes, strye);
}
```

Операции над строками

Над строками определены следующие операции:

- присваивание (=);
- две операции проверки эквивалентности (= =) и (!=);
- конкатенация или сцепление строк (+);
- взятие индекса ([]).

Неизменяемый класс string

- В языке C# существует понятие неизменяемый (immutable) класс. Для такого класса невозможно изменить значение объекта при вызове его методов. Динамические методы могут создавать новый объект, но не могут изменить значение существующего объекта.

Метод	Описание
Empty	Возвращается пустая строка. Свойство со статусом read only
Compare	Сравнение двух строк. Метод перегружен. Реализации метода позволяют сравнивать как строки, так и подстроки. При этом можно учитывать или не учитывать регистр, особенности национального форматирования дат, чисел и т.д.
CompareOrdinal	Сравнение двух строк. Метод перегружен. Реализации метода позволяют сравнивать как строки, так и подстроки. Сравниваются коды символов
Concat	Конкатенация строк. Метод перегружен, допускает сцепление произвольного числа строк
Copy	Создается копия строки
Format	Выполняет форматирование в соответствии с заданными спецификациями формата. Ниже приведено более полное описание метода
Intern, IsIntern	Отыскивается и возвращается ссылка на строку, если таковая уже хранится во внутреннем пуле данных. Если же строки нет, то первый из методов добавляет строку во внутренний пул, второй - возвращает null. Методы применяются обычно тогда, когда строка создается с использованием построителя строк - класса StringBuilder

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
Join	Конкатенация массива строк в единую строку. При конкатенации между элементами массива вставляются разделители. Операция, заданная методом Join, является обратной к операции, заданной методом Split. Последний является динамическим методом и, используя разделители, осуществляет разделение строки на элементы

Примеры работы

- *// Сравним первые две строки*
- **string** s1 = "это строка";
- **string** s2 = "это текст, а это строка";
- **if** (String.CompareOrdinal(s1, s2) != 0) Console.WriteLine("Строки s1 и s2 не равны");
- **if** (String.Compare(s1, 0, s2, 13, 10, **true**) == 0) Console.WriteLine("При этом в них есть одинаковый текст");
- *// Конкатенация строк* Console.WriteLine(String.Concat("\n" + "Один, два ","три, четыре"));
- *// Поиск в строке // Первое вхождение подстроки*
- **if** (s2.IndexOf("это") != -1) Console.WriteLine("Слово \" это\" найдено в строке, оно "+ "находится на: {0} позиции", s2.IndexOf("это"));

Примеры работы

- *// Последнее вхождение подстроки* **if** (s2.LastIndexOf("это") != -1) Console.WriteLine("Последнее вхождение слова \"это\" находится " + "на {0} позиции", s2.LastIndexOf("это"));
- *// Поиск из массива символов*
- **char**[] myCh = {'Ы','х','т'};
- **if** (s2.IndexOfAny(myCh) != -1) Console.WriteLine("Один из символов из массива ch "+ "найден в текущей строке на позиции {0}", s2.IndexOfAny(myCh)); /

Примеры работы

- *// Определяем начинается ли строка с заданной подстроки*
- **if** (s2.StartsWith("это текст") == **true**) Console.WriteLine("Подстрока найдена!");
- *// Определяем содержится ли в строке подстрока*
- *// на примере определения ОС пользователя* **string** myOS = Environment.OSVersion.ToString();
- **if** (myOS.Contains("NT 5.1")) Console.WriteLine("Ваша операционная система Windows XP");

Класс StringBuilder. Конструкторы

- public StringBuilder (string str, int cap). Параметр str задает строку инициализации, cap - емкость объекта;
- public StringBuilder (int curcap, int maxcap). Параметры curcap и maxcap задают начальную и максимальную емкость объекта;
- public StringBuilder (string str, int start, int len, int cap). Параметры str, start, len задают строку инициализации, cap - емкость объекта.

Операции над строками

Над строками этого класса определены практически те же операции с той же семантикой, что и над строками класса String:

- присваивание (=);
- две операции проверки эквивалентности (= =) и (!=);
- взятие индекса ([]).
- Операция конкатенации (+) не определена над строками класса StringBuilder, ее роль играет метод Append, дописывающий новую строку в хвост уже существующей.

Особенность класса StringBuilder

- Со строкой этого класса можно работать как с массивом, но, в отличие от класса String, в нем допускается изменение символа.

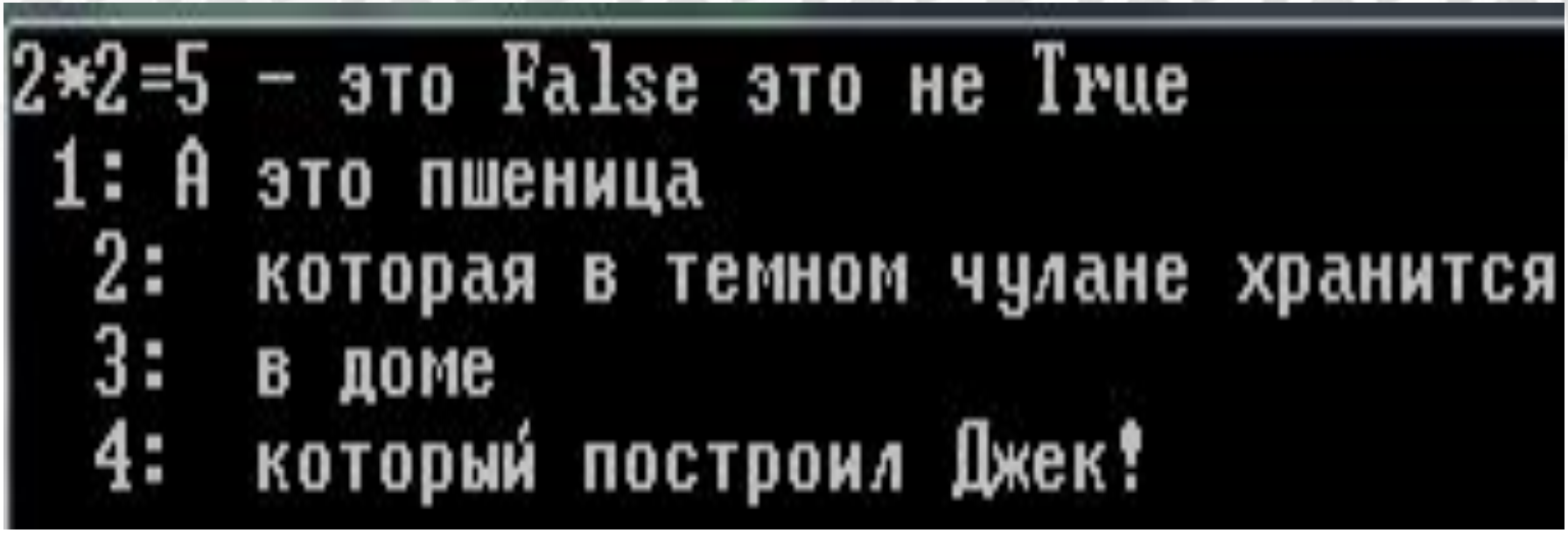
Методы StringBuilder

- public StringBuilder Append (<объект>). К строке, вызвавшей метод, присоединяется строка, полученная из объекта, который передан методу в качестве параметра.
- public StringBuilder Insert (int location,<объект>). Метод вставляет строку, полученную из объекта, в позицию, указанную параметром location.
- public StringBuilder Remove (int start, int len). Метод удаляет подстроку длины len, начинающуюся с позиции start;
- public StringBuilder Replace (string str1,string str2). Все вхождения подстроки str1 заменяются на строку str2;
- public StringBuilder AppendFormat (<строка форматов>, <объекты>). Метод является комбинацией метода Format класса String и метода Append.

Примеры работы

- //Методы Insert, Append, AppendFormat
  StringBuilder strbuild = new StringBuilder();
  string str = "это это не ";
  strbuild.Append(str); strbuild.Append(true);
  strbuild.Insert(4,false); strbuild.Insert(0,"2*2=5 - ");
  Console.WriteLine(strbuild);
  string txt = "А это пшеница, которая в темном
  чулане
  хранится," +" в доме, который построил Джек!";
  StringBuilder txtbuild = new StringBuilder();
  int num =1;
  foreach(string sub in txt.Split(','))
  {
  txtbuild.AppendFormat(" {0}: {1}\n ", num++,sub);
  }
  str = txtbuild.ToString();
  Console.WriteLine(str);

Примеры работы



Емкость буфера

- Каждый экземпляр строки класса StringBuilder имеет буфер, в котором хранится строка. Объем буфера - его емкость - может меняться в процессе работы со строкой.
- свойство Capacity - возвращает или устанавливает текущую емкость буфера;
- свойство MaxCapacity - возвращает максимальную емкость буфера. Результат один и тот же для всех экземпляров класса;
- метод int EnsureCapacity (int capacity) - позволяет уменьшить емкость буфера.

Регулярные выражения

- Основа обработки текста с помощью регулярных выражений — это подсистема обработки регулярных выражений, представленная в платформе .NET Framework объектом System.Text.RegularExpressions.Regex.

Регулярные выражения

Минимальный набор сведений, который требуется предоставить подсистеме обработки регулярных выражений для обработки текста с помощью регулярных выражений, сводится к двум вещам.

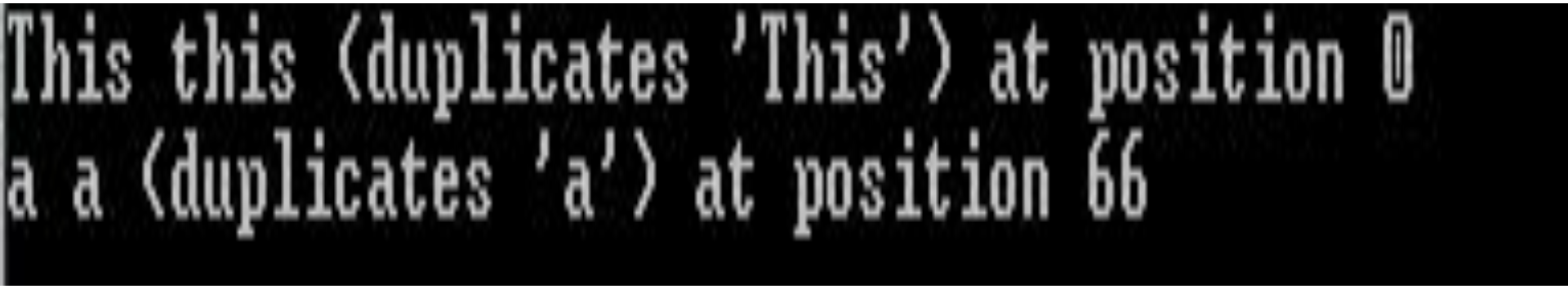
- Шаблон регулярного выражения, который требуется найти в тексте.(pattern)
- Текст, который требуется проанализировать с помощью шаблона регулярного выражения.

Методы класса Regex

- Определить, встречается ли во входном тексте шаблон регулярного выражения, можно путем вызова метода IsMatch.
- Извлечь из текста одно или все вхождения, соответствующие шаблону регулярного выражения, можно путем вызова метода Match или Matches.
- Заменить текст, соответствующий шаблону регулярного выражения, можно путем вызова метода Replace.

Примеры работы

- public static void Main() {
- string pattern = @"\b(\w+?)\s\1\b";
- string input = "This this is a nice day. What about this? This tastes good. I saw a a dog.";
- foreach (Match match in Regex.Matches(input, pattern, RegexOptions.IgnoreCase))
- Console.WriteLine("{0} (duplicates '{1}') at position {2}", match.Value, match.Groups[1].Value, match.Index); }



Примеры работы

\b	Начало на границе слова.
(\w+)	Совпадение с одним или несколькими символами слова. Вместе они формируют группу, на которую можно сослаться, указав обозначение \1.
\s	Соответствует пробелу.
\1	Соответствует подстроке, совпадающей с группой \1.
\b	Соответствует границе слова.

Аккредитационная задача

- Используя регулярные выражения, напишите следующее приложение: дан текст, имеющий структуру «Фамилия И.О. – улица – номер дома – квартира – номер телефона». Вывести на экран фамилии всех абонентов, проживающих на улице Реввоенсовета.

```
StringBuilder text = new StringBuilder();
text.Append("Стринг И.О. – улица – 1 – 43 – 213123\n");
text.Append("СтрингБилдер И.О. – улица – 2 – 12312 – 32423\n");
text.Append("СтрингБилдер И.О. – Реввоенсовета – 2 – 12312 – 32423\n");
text.Append("Чар А.В. – Реввоенсовета – 2 – 12312 – 32423\n");
text.Append("СтрингБилдер И.О. – улица – 2 – 12312 – 32423\n");
string pattern =  @"^(\w+?)\s(.*?)– Реввоенсовета";
Regex reg = new Regex(pattern,RegexOptions.Multiline);
MatchCollection mc = reg.Matches(text.ToString());
foreach (Match match in mc)
 Console.WriteLine("ФИО:{0}", match.Groups[1]);
 Console.ReadKey();
```