Строки и символы

Задание 2 (Класс Regex)

Используем:

- Лекция Регулярные выражения: сайт дисциплины
- Класс Regex: https://habr.com/ru/company/otus/blog/469989/
- https://docs.microsoft.com/ru-ru/dotnet/standard/base-types/regular-expressions
- https://professorweb.ru/my/csharp/charp_theory/level4/4_10.php

Начало работы с регулярными выражениями

Для того, чтобы работать с регулярными выражениями необходимо подключить в начале программы пространство имен using System.Text.RegularExpressions; В C# работу с регулярными выражениями предоставляет класс Regex. Создание регулярного выражения имеет следующий вид:

```
Regex myReg = new Regex([шаблон]);
```

Здесь [шаблон] – это строка содержащая символы и спецсимволы. У Regex также есть и второй конструктор, который принимает дополнительный параметр – опции поиска (пример в конце теории).

пример:

```
static void Main(string[] args)
{
  string data1 = "Петр, Андрей, Николай";
  string data2 = "Петр, Андрей, Александр";
  Regex myReg = new Regex("Николай"); // создание регулярного выражения
  Console.WriteLine(myReg.IsMatch(data1)); // True
  Console.WriteLine(myReg.IsMatch(data2)); // False
  Console.ReadKey();
}
```

Здесь в качестве шаблона выступает однозначная строка "Николай". Дальше был использован метод IsMatch, который проверят, содержит ли заданная строка (data1, data2) подстроку соответствующую шаблону.

Методы класса Regex

IsMatch – проверяет содержит ли строка хотя бы одну подстроку соответствующую шаблону регулярного выражения. Работа этого метода показана в примере выше.

Match – возвращает первую подстроку, соответствующую шаблону, в виде объекта класса Match. Класс Match предоставляет различную информацию о подстроке – длину, индекс, само значение и другое.

```
static void Main(string[] args)
{
  string data1 = "Петр, Андрей, Николай";
  Regex myReg = new Regex("Николай");
  Match match = myReg.Match(data1);
  Console.WriteLine(match.Value); // "Николай"
  Console.WriteLine(match.Index); // 14
  Console.ReadKey();
}
```

Matches – возвращает все подстроки соответствующие шаблону в виде коллекции типа MatchCollection. Каждый элемент этой коллекции типа Match.

```
static void Main(string[] args)
{
  string data1 = "Петр, Николай, Андрей, Николай";
  Regex myReg = new Regex("Николай");
  MatchCollection matches = myReg.Matches(data1);
  Console.WriteLine(matches.Count); // 2
  foreach (Match m in matches)
    Console.WriteLine(m.Value); //вывод всех подстрок "Николай"
```

```
    Console.ReadKey();
}
```

Replace – возвращает строку, в которой заменены все подстроки, соответствующие шаблону, новой строкой:

```
static void Main(string[] args)
{
    string data1 = "Петр, Николай, Андрей, Николай";
    Regex myReg = new Regex("Николай");
    data1 = myReg.Replace(data1, "Максим");
    Console.WriteLine(data1); //"Петр, Максим, Андрей, Максим"
    Console.ReadKey();
}
```

Split – возвращает массив строк, полученный в результате разделения входящей строки в местах соответствия шаблону регулярного выражения:

```
static void Main(string[] args)
{
    string data1 = "Петр,Николай,Андрей,Николай";
    Regex myReg = new Regex(",");
    string[] names = myReg.Split(data1); // массив имен
    Console.ReadKey();
}
```

Специальные символы

В примерах выше рассматривались очень простые, однозначные регулярные выражения без использования спецсимволов.

Классы символов

Обозначение	Описание	Шаблон	Соответствие
[группа_символов]	Любой из перечисленных в скобках символов. Используя тире можно указать диапазон символов, например, [a-f] - то же самое, что [abcdef]	[abc]	«a» в «and»
[^группа_символов]	Любой символ, кроме перечисленных в скобках	[^abc]	«n», «d» в «and»
\d	Цифра. Эквивалентно [0-9]	\d	«1» в «data1»
\D	Любой символ, кроме цифр. Эквивалентно [^0-9]	\D	«y» в «2014y»
\w	Цифра, буква (латинский алфавит) или знак подчеркивания. Эквивалентно [0-9a-zA-Z_]	\w	«1», «5», «c» в «1.5c»
\W	Любой символ, кроме цифр, букв (латинский алфавит) и знака подчеркивания. Эквивалентно [^0-9a-zA-Z_]	\W	«.» в «1.5c»
\s	Пробельный символ (пробел, табуляция, перевод строки и т. п.)	\s	« » в «c sharp»
\S	Любой символ, кроме пробельных	\S	«c» «s» «h» «a» «r» «p» в «c sharp»
.	Любой символ, кроме перевода строки. Для поиска любого символа, включая перевод строки, можно использовать конструкцию [\s\S]	c.harp	«csharp» в «mycsharp»

Символы повторения

Обозначение	Описание	Шаблон	Соответствие
*	Соответствует предыдущему элементу ноль или более раз	\d*.	«a», «1b», «23c » в «a1b23c»
+	Соответствует предыдущему элементу один или более раз	\d+.	«1b», «23c » в «a1b23c»
?	Соответствует предыдущему элементу ноль или один раз	\d?\D	«a», «1b», «3c» в «a1b23c»
{n}	Соответствует предыдущему элементу, который повторяется ровно n раз	\d{2}	«43», «54», «82» в «2,43,546,82»
{n,}	Соответствует предыдущему элементу, который повторяется минимум n раз	\d{2,}	«43», «546», «82» в «2,43,546,82»
{n,m}	Соответствует предыдущему элементу, который повторяется минимум n раз и максимум m	\d{2,}	«43», «546», «821» в «2,43,546,8212»

Символы привязки

Обозначение	Описание	Шаблон	Соответствие
^	Соответствие должно находиться в начале строки	^\d{2}	«32» в «32,43,54»
$	Соответствие должно находиться в конце строки или до символа \n при многострочном поиске	\d{2}$	«54» в «32,43,54»
\b	Соответствие должно находиться на границе алфавитно-цифрового символа (\w) и не алфавитно-цифрового (\W)	\b\d{2}	«32», «54» в «32 a43 54»
\B	Соответствие не должно находиться на границе	\B\d{2}	«43» в «32 a43 54»
\G	Соответствие должно находиться на позиции конца предыдущего соответствия	\G\d	«3», «2», «4» в «324.758»

Символы выбора

Обозначение	Описание	Шаблон	Соответствие
|	Работает как логическое «ИЛИ» - соответствует первому и/или второму шаблону	one|two	«one», «two» в «one two three»
(группа_символов)	Группирует набор символов в единое целое для которого дальше могут использоваться + * ? и т.д. Каждой такой группе назначается порядковый номер слева направо начиная с 1. По этому номеру можно ссылаться на группу \номер_группы	(one)\1	«oneone» в «oneone onetwoone»
(?:группа_символов)	Та же группировка только без назначения номера группы	(?:one){2}	«oneone» в «oneone onetwoone»

Другие символы

Обозначение	Описание	Шаблон	Соответствие
\t	Символ табуляции	\t	
\v	Символ вертикальной табуляции	\v	
\r	Символ возврата каретки	\r	
\n	Символ перевода строки	\n	
\f	Символ перевода страницы	\f	
\	Символ, который позволяет экранировать специальные символы, чтобы те воспринимались буквально. Например, чтобы было соответствие символу звёздочки, шаблон будет выглядеть так *	\d\.\d	«1.1», «1.2» в «1.1 1.2»

В этих таблицах представлены далеко не все элементы языка регулярных выражений, но их вполне достаточно для больших возможностей.

пример (корректность электронного адреса):

```
static void Main(string[] args)
{
  Regex myReg = new Regex(@"[A-Za-z]+[\.A-Za-z0-9_-]*[A-Za-z0-9]+@[A-Za-z]+\.[A-Za-z]+");
  Console.WriteLine(myReg.IsMatch("email@email.com")); // True
  Console.WriteLine(myReg.IsMatch("email@email")); // False
  Console.WriteLine(myReg.IsMatch("@email.com")); // False
  Console.ReadKey();
}
```

Здесь перед началом строки регулярного выражения стоит символ «@» который указывает компилятору воспринимать все символы буквально. Это необходимо, чтобы корректно воспринимался символ «\».

Параметры поиска

Использование второго конструктора Regex, который принимает в качестве второго аргумента значение перечисления RegexOptions. В этом перечислении есть следующие значения:
IgnoreCase – игнорирование регистра при поиске. Находит соответствия независимо прописными или строчными буквами в строке написано слово;
RightToLeft – поиск будет выполнен справа налево, а не слева направо;
Multiline – многострочный режим поиска. Меняет работу спецсимволов «^» и «$» так, что они соответствуют началу и концу каждой строки, а не только началу и концу целой строки;
Singleline – однострочный режим поиска;
CultureInvariant - игнорирование национальных установок строки;
ExplicitCapture – обеспечивается поиск только буквальных соответствий;
Compiled – регулярное выражение компилируется в сборку, что делает более быстрым его исполнение но увеличивает время запуска;
IgnorePatternWhitespace – игнорирует в шаблоне все неэкранированные пробелы. С этим параметром шаблон «a b» будет аналогичным шаблону «ab»;
None – использовать поиск по умолчанию.

Пример использование параметра поиска (игнорирование регистра):

```
static void Main(string[] args)
{
  string data = "nikolay, sergey, oleg";
  Regex myRegIgnoreCase = new Regex(@"Sergey", RegexOptions.IgnoreCase);
  Regex myReg = new Regex(@"Sergey");
  Console.WriteLine(myRegIgnoreCase.IsMatch(data)); // True
  Console.WriteLine(myReg.IsMatch(data)); // False
  Console.ReadKey();
}
```

Если необходимо установить несколько параметров, тогда они разделяются оператором поразрядного «ИЛИ» – «|»

```
Regex myReg = new Regex(@"Sergey", RegexOptions.IgnoreCase | RegexOptions.IgnorePatternWhitespace);
```

Пример. Создадим метод, который будет принимать ссылку, а возвращать только доменное имя:

```
public static string GetDomain(string url)
{
  Regex re = new Regex("http://", RegexOptions.IgnoreCase);
  url = re.Replace(url, ""); // удаляем часть http://

  Regex reWww = new Regex(@"www\.", RegexOptions.IgnoreCase);
  url = reWww.Replace(url, ""); //удаляем часть www.

  int end = url.IndexOf("/");
  if (end != -1)
    url = url.Substring(0, end);
  return url;
}
static void Main(string[] args)
{
  string url1 = "http://mycsharp.ru/post/33/2013_10_19_virtualnye_metody_v_si-sharp_pereopredelenie_metodov.html";
  string url2 = "http://www.mycsharp.ru/post/33/2013_10_19_virtualnye_metody_v_si-sharp_pereopredelenie_metodov.html";
  Console.WriteLine(GetDomain(url1)); // mycsharp.ru
  Console.WriteLine(GetDomain(url2)); // mycsharp.ru
  Console.ReadKey();
}
```

1. Поиск строк в файле

В заданном текстовом файле найти указанные строчки и выдать позиции необходимых подстрок:

Вариант 1. Найдите строчки, в которых есть натуральные числа (возможно, окружённые буквами);
Вариант 2. Найдите строчки, в которых есть «слова», написанные капсом (то есть строго заглавными), возможно внутри настоящих слов (аааБББввв);
Вариант 3. Найдите строчки, в которых есть слова, в которых есть русская буква, а когда-нибудь за ней цифра;
Вариант 4. Найдите строчки, в которых есть слова, начинающиеся с русской или латинской большой буквы (`\b` — граница слова);
Вариант 5. Найдите строчки, в которых есть слова, которые начинаются на гласную (`\b` — граница слова);;
Вариант 6. Найдите строчки, в которых есть натуральные числа, не находящиеся внутри или на границе слова;
Вариант 7. Найдите строчки, в которых есть символ `*` (`.` — это точно не конец строки!);
Вариант 8. Найдите строчки, в которых есть открывающая и когда-нибудь потом закрывающая скобки;
Вариант 9. Найдите весь кусок оглавления в html-тексте (вместе с тегами);
Вариант 10. Найдите только текстовую часть оглавления в html-тексте, без тегов;

2. Фильтрация строк в файле

Задан текстовый файл, содержащий некоторое множество строк. Требуется вывести те из строк, которые удовлетворяют заданному критерию. В приведенных ниже задачах вам нужно написать регулярное выражение для описанного критерия. Должны быть выведены в точности строки, удовлетворяющие заданному критерию. Далее словом называется непустая последовательность символов, подходящих под шаблон «\w», ограниченная с двух сторон началом/концом строки или остальными символами («\W»). Подсказка: для работы со словами удобны также шаблоны «\b» и «\B». Под термином буква подразумевается символ, подходящий под шаблон «\w».

Вариант 1. Строки, содержащие «cat» в качестве подстроки два раза. Пример строк, которые подходят: «catcat», «cat and cat». Пример строк, которые не подходят: «catac», «cat», «ccaatt».
Вариант 2. Строки, содержащие «cat» в качестве слова. Пример строк, которые подходят: «cat», «catapult and cat», «catapult and cat and concatenate». Пример строк, которые не подходят: «catcat», «concat», «Cat».
Вариант 3. Строки, содержащие «cat» в качестве подстроки, игнорируйте регистр. Пример строк, которые подходят: «cat», «cat and cat», «Cat», «theCATisHERE». Пример строк, которые не подходят: «kat», «», «cot».
Вариант 4. Строки, содержащие две буквы «z», между которыми ровно три символа. Пример строк, которые подходят: «zabcz», «zzxzz». Пример строк, которые не подходят: «zzz», «zz», «zxz», «zzxzxxz».

Вариант 5. Строки, содержащие две буквы из множества {«x», «y», «z»}, между которыми от 5 до 17 символов. Пример строк, которые подходят: «xabcabcz», «zzzzzzzzzzzzzzzzzz». Пример строк, которые не подходят: «xx», «xyz», «zwzwwz».

Вариант 6. Строки, содержащие в качестве слова целое число. Пример строк, которые подходят: «Year is 2009.», «1 is a number», «3.1415 matches because . is not a word char». Пример строк, которые не подходят: «Not2Bad», «No digits here».

Вариант 7. Строки, содержащие обратный слеш. Пример строк, которые подходят: «\w denotes word character». Пример строк, которые не подходят: «No slashes here».

Вариант 8. Строки, содержащие слово внутри произвольного текста, не содержащего скобок, в скобках. Пример строк, которые подходят: «good (excellent) phrase», «good (too bad) phrase», «good ((recursive)) phrase». Пример строк, которые не подходят: «word () is not () in brackets», «bad (() recursive) phrase», «no brackets here».

Вариант 9. Строки, не содержащие ведущих или конечных пробельных символов. Пример строк, которые подходят: «Good string», «». Пример строк, которые не подходят: « bad string», «bad string », « very bad string ».

Вариант 10. Строки, содержащие слово, состоящее из двух равных частей (тандемный повтор). Пример строк, которые подходят: «blabla is a tandem repetition» «123123 is good too». Пример строк, которые не подходят: «go go», «aaa»,.

3. Преобразование строк в файле Задан текстовый файл, содержащий некоторое множество строк. Требуется преобразовать каждую из этих строк в соответствии с заданным правилом (нужно написать регулярное выражение для описанного преобразования) и вывести результат. Также вывести количество успешных замен.

Вариант 1. Поменять местами две первых слова в каждом предложении. Примеры замен: «this is a text.» → «is this a text.», «(This,) is also a text!» → «(is,) This also a text!».

Вариант 2. Поменять местами две первых буквы в каждом слове. Примеры замен: «this is a text» → «htis si a etxt».

Вариант 3. Заменить все вхождения двух одинаковых букв подряд на одну букву. Примеры замен: «attraction» → «atraction», «buzzzz» → «buzz».

Вариант 4. Заменить все вхождения нескольких одинаковых букв подряд на одну букву. Примеры замен: «attraction» → «atraction», «buzzzz» → «buz».

Вариант 5. Заменить все числа кратные 10 на их частное от деления на 10. В этой задаче на вход подаются числа, разделенные пробелами. Примеры замен: «1 2 10 12 20 123 239 566 12800» → «1 2 1 12 2 123 239 566 1280».

Вариант 6. Удалить символы после каждой открывающейся скобки до ближайшей закрывающейся, кроме первого символа. Примеры замен: «(word) outside (1 open (2 open)» → «(w) outside (1)».

Вариант 7. Будем называть хорошей подстроку, состоящую хотя бы из двух символов, если она начинается с буквы «a» и заканчивается буквой «a». Заменить все вхождения трех хороших строк подряд на строку «bad». При этом замена должна производиться как только соответствующая подстрока встретилась. Примеры замен: «abaacaada» ⟶ «bad», «abaacaadaa» → «bada».

Вариант 8. Заменить все вхождения нескольких одинаковых слов подряд на одно слово. Примеры замен: «весна весна пришла» ⟶ «весна пришла».

Вариант 9. Осуществить замену строк вида: Иван Иванович Иванов -> И.И. Иванов

Вариант 10. В заданном тексте удалить все заглавные буквы в начале слов. Примеры замен: «asd XFcvG Hytr» ⟶ «asd FcvG ytr».