Иллюстрации к курсу лекций по дисциплине «Программирование»

Введение в **регулярные выражения**

1. Общие сведения

Обозначение в DOS всех файлов с одним именем:
«Итоговая_ведомость.*»

Традиционные задачи:

- Проверка наличия подстрок.
- Проверка соответствия строки.
- Выбор из строки подстрок.
- Замены подстрок.

Регулярные выражения в .NET

```
using System.Text.RegularExpressions;
```

2. Язык регулярных выражений

Элементы регулярного выражения:

- **терминальные символы** (объекты формальной грамматики, имеющие в нём конкретное неизменяемое значение и являющиеся элементом построения слов данного языка, обобщение понятия “буквы”);
- **метасимволы** (объекты, обозначающий какую-либо сущность языка и не имеющие конкретного символьного значения).

Прототип метода поиска подстроки:

public static **Match** Match(string str, string reg)

- str - исследуемая строка;
- reg - регулярное выражение.

3. Свойства класса Match

- Index – самая левая позиция в строке, начиная с которой размещена найденная подстрока.
- Length – длина найденной подстроки.
- Value – найденная подстрока.
- Success – логическое значение (успех поиска).

4. Примеры

```
Match result = Regex.Match("together", @"th");
Console.WriteLine(result.Success);       // True
Console.WriteLine(result.Value);         // th
Console.WriteLine(result.Index);         // 4
Console.WriteLine(result.Length);        // 2
Console.WriteLine(result.ToString());    // th
Console.WriteLine(Regex.Match("format",
    @"\u0066\u006F\u0072"));     // for

// \u0066 = 'f', \u006F = 'o', \u0072 = 'r'.
```

5. Экранирование метасимволов

\	→	\\	*	→	*
+	→	\+	?	→	\?
|	→	\\|	{	→	\{
}	→	\}	[	→	\[
]	→	\]	(	→	\(
)	→	\)	^	→	\^
$	→	\$	.	→	\.
#	→	\#			

6. Подмножества символов

. – любой одиночный символ

Примеры:

```
Console.WriteLine(
  Regex.Match("шило и мыло", @"..ло")); // шило

Console.WriteLine(
  Regex.Match("11.22.33.44.55.66", @"4\.5"));// 4.5
```

7. Метод Matches

Прототип:

public static **MatchCollection** Matches(string str,
string reg)

Пример:

```
foreach (Match mat in
  Regex.Matches("11.22.33.44.55.66", @".\.."))
   Console.Write(mat.Value + "  ");


  // 1.2  2.3  3.4  4.5  5.6
```

8. Метасимвол альтернативного выбора

MatchCollection – коллекция элементов типа Match

Пример подмножества шестнадцатеричных цифр:

[0-9a-fA-F]

```
// \u0020 - [пробел]
MatchCollection group = Regex.Matches
        ("кот кит кат кто ком", @"\u0020к[ио].");
foreach (Match mat in group)
    Console.WriteLine(mat.Value);
```

Результаты работы:

кит

ком

9. Обозначения подмножеств

\d - десятичная цифра ➔ [0-9]
\w - символ идентификатора, ➔ [a-zA-Z_0-9]
\s - пробельный символ ➔ [\n\r\t\f]

\p{*категория*}.

10. Категории символов

\p{L} – любые буквы любого языка,
\p{Lu} – буквы в верхнем регистре (прописные),
\p{Ll} – буквы в нижнем регистре (строчные),
\p{N} – цифры,
\p{P} – любые пунктуационные знаки,
\p{M} – диакритические знаки (ë, î, ö),
\p{S} – символы отличные от пробельных,
\p{Z} – пробелы, невидимые разделители,
\p{C} – управляющие символы.

11. «Отрицательные» подмножества

Пример:
[^0-4] – запрещены цифры от 0 до 4

\D ➔ [^0-9]
\W ➔ [^a-zA-Z_0-9]
\S ➔ [^\n\r\t\f]
\P{*категория*} - любой символ, кроме символа, принадлежащего категории.

12. Директивы нулевой ширины

^ - Соответствие в начале строки.

$ - Соответствие в конце строки или перед символом **\n**.

\A - Соответствие в начале строки.

\z - Соответствие в конце строки.

\G - Соответствие в конце другого соответствия.

\b - Соответствие на границе между **\w** и **\W** .

\B - Соответствие не на границе **\b**.

13. Примеры

```
MatchCollection res =
 Regex.Matches("кот кит кат кто ком", @"к[^та].$");
foreach (Match mat in res)
    Console.Write(mat.Value + "  ");    // ком
```

```
res =
 Regex.Matches("кот кит кат кто ком", @"^к[^та].");
foreach (Match mat in res)
    Console.Write(mat.Value + "  ");    // кот
```

14. Квантификаторы

"zo*"	соответствует "z", "zo" и "zooo".
"zo+"	соответствует "zo" и "zooo" (не "z").
"do(es)?"	соответствует "do" и "does" (не "doeses").
{n}	"o{2}" соответствует "oo" (не "o" / "ooo").
{n,}	"o{2,}" соответствует "ooo" (не "o"). "o{1,}" == "o+", "o{0,}" == "o*".
{n,m}	m и n — неотрицательные целые числа, где n <= m. "o{0,1}" == "o?".

15. Пример – выделение слов из текста

```
string line = " 38 попугаев и другие    персонажи";
foreach (Match mt in
      Regex.Matches(line, @"\b\w+\b"))
          Console.WriteLine(mt.Value);
```

Результат:

38

попугаев

и

другие

персонажи

Пример с объектом класса Regex

```
Regex key = new Regex(@"\w{6,}");
string citato = "Conditio sine qua non – непременное условие";
MatchCollection words = key.Matches(citato);
foreach (Match mt in words)
    Console.WriteLine("Слово: {0}, длина: {1}, позиция: {2}",
        mt.Value, mt.Length, mt.Index);
```

Результат:

Слово: Conditio, длина: 8, позиция: 0
Слово: непременное, длина: 11, позиция: 24
Слово: условие, длина: 7, позиция: 36

17. «Жадные» и «ленивые» квантификаторы

// * – жадный квантификатор:

```
Console.WriteLine(Regex.Match("сорок сороков", @".*к"));
```

Результат поиска: «сорок сорок»

// *? – ленивый квантификатор:

```
Console.WriteLine(Regex.Match("сорок сороков", @".*?к"));
```

Результат поиска: «сорок»

```
String text = "рациональная <b>дробь</b> – " +
              "это упорядоченная пара <b>целых</b> чисел";
Regex.Match(text, @"<b>.*?</b>");
```

Результат поиска: «дробь»

Для регулярного выражения ,@".*"

Результат поиска – длинная строка:

«дробь – это упорядоченная пара целых»

18. Поиск с просмотром

```
string URL =
@"http:\\www.ecom.ru\tricks\tric01\nature.htm";
Console.WriteLine(Regex.Match(URL, @"\\\\.*?\\"));
// «Неудачный» результат:  \\www.ecom.ru\
```

Метасимволы просмотра:

(?<=*выражение*) – просмотр назад

(?=*выражение*) – просмотр вперед

(?<!*выражение*) – просмотр назад с отрицанием

(?!*выражение*) – просмотр вперед с отрицанием

```
Console.WriteLine(Regex.Match(URL,
      @"(?<=\\\\).*?(?=\\)"));
// Результат: www.ecom.ru
```

19. Группы регулярного выражения

Пример анализа телефонного номера:

```
string phone = @"101-421-232-44-57";
string reg = @"(\d\d\d)-(\d\d\d)-(\d\d\d-\d\d-\d\d)";
Match memb = Regex.Match(phone, reg);
Console.WriteLine("Полный номер: " + memb.Groups[0]);
Console.WriteLine("Код страны: " + memb.Groups[1]);
Console.WriteLine("Код города: " + memb.Groups[2]);
Console.WriteLine("Номер телефона: " + memb.Groups[3]);
```

Результат выполнения фрагмента программы:

Полный номер: 101-421-232-44-57
Код страны: 101
Код города: 421
Номер телефона: 232-44-57

20. Именованные группы

Именование:

(?<*имя_группы*>....)

(?'*имя_группы*'....)

Обращение:

\k<*имя_группы*>

\k'*имя_группы*'

21. Пример с обратной ссылкой

```
string sentence = @"Penny saved is a Penny got.";
string pattern = @"(?'first'\w+)(.*)\k'first'";
Match result = Regex.Match(sentence, pattern);
Console.WriteLine(result.Length);
Console.WriteLine(result.Groups[0]);
Console.WriteLine(result.Groups[1]);
Console.WriteLine(result.Groups["first"]);
```

Результаты выполнения фрагмента программы:

22
Penny saved is a Penny
 saved is a
Penny

22. Выделить HTML-элемент

```
string regFind =
@"<(?'tag'\w+).*?>(?'text'.*?)</\k'tag'>";
string str = @"<script
language=""JavaScript"">Content!!!</script>";

Match m = Regex.Match(str, regFind);
Console.WriteLine(m.Groups[0]);
Console.WriteLine(m.Groups["tag"]);
Console.WriteLine(m.Groups["text"]);
```

Результаты:

<script language="JavaScript">Content!!!</script>
script
Content!!!

23. Замены в тексте

Прототип (заголовок) метода:

```
public static string Replace( string input, string pattern,
        string replacement )
```

Пример:

```
string find = @"h3>";
string change = @"h5>";
Console.WriteLine(Regex.Replace(
    "Начало <h3>Оглавление</h3> Конец",
            find, change));
```

Результат:

Начало <h5>Оглавление</h5> Конец

24. Обращения к группам в шаблоне подстановки

$i где i=0,1,2,… или ${*имя_группы*}

```
string line = "кот кит кат кто ком";
Console.WriteLine(Regex.Replace(line,
      @"к[ои]т", @"'$0'"));
```

Результат:

'кот' 'кит' кат кто ком

25. Делегат-параметр метода Replace

Прототип (заголовок) метода:

public static string Replace(string input, string pattern,
MatchEvaluator evaluator)

Объявление (сигнатура) делегата:

public delegate string **MatchEvaluator**(**Match** match)

Пример:

```
Console.WriteLine(Regex.Replace("3 1 0 6 2",
@"\d",
    r => (r.Value == "0" ? "нуль" :
          r.Value == "1" ? "один" : "NaN")));
```

Результат:

NaN один нуль NaN NaN

26. Деление текста

Прототип метода:
public static string [ ] Split (string input, string pattern)

Выделить слова предложения:

```
string line = "Veni,  vidi,  vici.";
Console.WriteLine(line);
foreach (string s in Regex.Split(line,
                                 @"[,.]\s*"))
    Console.Write(s + " ");
```

Результат:
Veni vidi vici