Декларативно програмиране в процедурните езици

Регулярни изрази

Владислав Ненчев

Софийски Университет “Св. Климент Охридски”
Факултет по Математика и Информатика
Катедра по Математическа Логика и Приложенията ѝ

21 Октомври 2016

Регулярни изрази - математическата дефиниция

.	-	конкатенация (обикновено на се пише)
$+$ или $\mid$	-	обединение на изрази/езици
$*$	-	звезда на Клини
$+$	-	дефинира се $R^+ = R^*.R$

Регулярни изрази - математическата дефиниция

.	-	конкатенация (обикновено на се пише)
+ или |	-	обединение на изрази/езици
*	-	звезда на Клини
+	-	дефинира се $R^+ = R^*.R$

$$0 + (1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9)(0 + 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9)^*$$

$$(a + \cdots + 9)^+(.(a + \cdots + 9)^+)^* @ (a + \cdots + 9)^+(.(a + \cdots + 9)^+)^+$$

$$(a + \cdots + Z)((a + \cdots + 9 + \_)^{5} + \cdots + (a + \cdots + 9 + \_)^{15})$$

$$\mathcal{RE}(\mathcal{A}(\Sigma^*(.+\cdots+?)\Sigma^*) \cap \mathcal{A}(\Sigma^*(0+\cdots+9)\Sigma^*) \cap \mathcal{A}(\Sigma^*(a+\cdots+Z)\Sigma^*) \cap \mathcal{A}((a+\cdots+?)^{8}(a+\cdots+?)^*))$$

$$((01+\cdots+31)/(01+\cdots+12)+(01+\cdots+30)/(04+\cdots+11)+01/02+\cdots+29/02)/(1+2)(0+\cdots+9)^{3}$$

Регулярни изрази в програмните езици

\d, \D	-	числови и нечислови символи
\s, \S	-	бели пространства и останали
\w, \W	-	символи за думи и останали
\b, \B	-	граница на дума и останали
^, $, \A, \z	-	начало и край на текст/ред
		(без \A и \z в JavaScript; \Z в Python)
\t, \n, \r, \v	-	специални символи
[ ], [^]	-	обвхати от символи и останали
a−z, A−Z, 0−9	-	специфични обвхати
.	-	произволен символ
|	-	алтернативи (обединение)
()	-	група (за разпознаване и замяна)
$1, $2, ...	-	идентификации за замяна
		(\1, \2, ... в Python)
\	-	символ за escape-ване

Регулярни изрази в програмните езици II

$*$	-	произволен брой символи
$+$	-	положителен брой символи
?	-	нула или един символи
{n}	-	точно n символа
{n,m}	-	поне n и най-много m символа
{n,}	-	поне n символа
$*$?, $+$?, ??, {n}?	-	минималистични варианти (в
{n,m}?, {n,}?		JavaScript и Python $*$? работи различно)

Регулярни изрази в програмните езици II

$*$	-	произволен брой символи
$+$	-	положителен брой символи
?	-	нула или един символи
{n}	-	точно n символа
{n,m}	-	поне n и най-много m символа
{n,}	-	поне n символа
$*$?, $+$?, ??, {n}?	-	минималистични варианти (в
{n,m}?, {n,}?		JavaScript и Python $*$? работи различно)

Без ^, $, \A и \z се разпознават под-низове.

Регулярни изрази в програмните езици II

$*$	-	произволен брой символи
$+$	-	положителен брой символи
?	-	нула или един символи
{n}	-	точно n символа
{n,m}	-	поне n и най-много m символа
{n,}	-	поне n символа
$*$?, $+$?, ??, {n}?	-	минималистични варианти (в
{n,m}?, {n,}?		JavaScript и Python $*$? работи различно)

Без ^, $, \A и \z се разпознават под-низове.
Следващи разпознавания стават след последното разпознаване (да се използва ?= за намиране на всички разпознавания).

Регулярни изрази в C#

Regex.IsMatch	-	проверява за съответствие
Regex.Match	-	намира първо съответствие (за следващи съответствия се използва Match.NextMatch)
Regex.Matches	-	намира всички съответствия
Regex.Replace	-	извършва замени

Регулярни изрази в C#

Regex.IsMatch	-	проверява за съответствие
Regex.Match	-	намира първо съответствие (за следващи съответствия се използва Match.NextMatch)
Regex.Matches	-	намира всички съответствия
Regex.Replace	-	извършва замени

```
bool ValidateNumber(string input) =>
  Regex.IsMatch(input, @"^0\$|^[1-9][0-9]*$");
```

```
bool ValidateEMail(string input) =>
  Regex.IsMatch(input,
    @"^[_\-\w]+(\.[_\-\w]+)*@[_\-\w]+(\.[_\-\w]+)+$");
```

Регулярни изрази в Java

Pattern.matches	-	проверява за ПЪЛНО съответствие
String.matches	-	проверява за ПЪЛНО съответствие
Matcher.find	-	намира съответствия последователно
Matcher.replaceAll	-	извършва замени
String.replaceAll	-	извършва замени

Регулярни изрази в Java

Pattern.matches	-	проверява за ПЪЛНО съответствие
String.matches	-	проверява за ПЪЛНО съответствие
Matcher.find	-	намира съответствия последователно
Matcher.replaceAll	-	извършва замени
String.replaceAll	-	извършва замени

```
static Boolean ValidateUsername(String username)
{
  return Pattern.
    matches("^[a-zA-Z][\\w_]{5,15}$", username);
}
static Boolean ValidateStrongPassword(String password)
{
  return Pattern.
    matches("^(?=.*[A-Z])(?=.*[a-z])(?=.*\\d)
    (?=.*[\\.;\\-_!\\?@])[\\w\\.;\\-_!\\?@]{8,}$",
        password);
 }
```

Регулярни изрази в JavaScript

RegExp.test	-	проверява за съответствие
RegExp.exec	-	намира съответствия последователно
String.match	-	намира всички съответствия
String.replace	-	извършва замени

Регулярни изрази в JavaScript

RegExp.test	-	проверява за съответствие
RegExp.exec	-	намира съответствия последователно
String.match	-	намира всички съответствия
String.replace	-	извършва замени

```
var exp = /(\d{2})\.(\d{2})\.(\d{4})/g;
var match = exp.exec(text);
while (match != null)
{
  if((match[1] == "25" || match[1] == "31") &&
    match[2] == "12" &&
    parseInt(match[3]) % 4 == 0)
      console.log("Christian holiday during
        leap year found: " + match[0] + "\n");
  match = exp.exec(text);
}
```

Регулярни изрази в Python

re.match	-	проверява за съответствие (в НАЧАЛОТО)
re. findall	-	намира всички съответствия
re. finditer	-	итератор на съответствията
re.sub	-	извършва замени

Регулярни изрази в Python

re.match	-	проверява за съответствие (в НАЧАЛОТО)
re. findall	-	намира всички съответствия
re. finditer	-	итератор на съответствията
re.sub	-	извършва замени

```
print(re.sub(r"\b(\w+)n't\b", r"\1 not", text))
```

```
for match in re.finditer(
  r"<(\w+)[^<>]*>\s*([^<>]*[^<>\s])\s*<", xml):
    print(match.group(1) + ": " + match.group(2))
```

Йода преводач

```
String translation = text.replaceAll("(?i)\\G(\\s*)
  (i|you|he|she|it|we|they|this|that)\\s+(\\w+)\\s+
    ([^\\.!\\?;]*)\\s*([\\.!\\?;])", "$1$4 $2 $3$5");
```

Йода преводач

```
String translation = text.replaceAll("(?i)\\G(\\s*)
  (i|you|he|she|it|we|they|this|that)\\s+(\\w+)\\s+
    ([^\\.!\\?;]*)\\s*([\\.!\\?;])", "$1$4 $2 $3$5");
```

"This is an example of Yoda translation, which is being done with a simple translator. I will translate a few sentences. They include some examples that change during the translation. Also, there are sentences that do not change. All sentences after them will not be translated. I hope that you had fun."

"an example of Yoda translation, which is being done with a simple translator This is. translate a few sentences I will. some examples that change during the translation They include. Also, there are sentences that do not change. All sentences after them will not be translated. I hope that you had fun."

Йода преводач (подобрение)

```
string translation = Regex.Replace(text,
  @"\G(\s*)(i|you|he|she|it|we|they|this|that)\s+
    (\w+)\s+([^\.!\?;]*)\s*([\.!\?;])",
  match =>
    match.Groups[1].Value +
    char.ToUpper(match.Groups[4].Value[0]) +
      match.Groups[4].Value.Substring(1) + " " +
    (match.Groups[2].Value.ToLower()  == "i" ? "I" :
      match.Groups[2].Value.ToLower()) + " " +
    match.Groups[3].Value +
    match.Groups[5].Value,
  RegexOptions.IgnoreCase);
```

Йода преводач (подобрение)

```
string translation = Regex.Replace(text,
  @"\G(\s*)(i|you|he|she|it|we|they|this|that)\s+
    (\w+)\s+([^\.!\?;]*)\s*([\.!\?;])",
  match =>
    match.Groups[1].Value +
    char.ToUpper(match.Groups[4].Value[0]) +
      match.Groups[4].Value.Substring(1) + " " +
    (match.Groups[2].Value.ToLower()  == "i" ? "I" :
      match.Groups[2].Value.ToLower()) + " " +
    match.Groups[3].Value +
    match.Groups[5].Value,
  RegexOptions.IgnoreCase);
```

"This is an example of Yoda translation. I will translate a few sentences."

"An example of Yoda translation this is. Translate a few sentences I will."

Упражнения

- Извличане на всички думи в текст.

Упражнения

- Извличане на всички думи в текст.
- Извличане на всички думи в текст, непосредствено след които се среща число.

Упражнения

- Извличане на всички думи в текст.
- Извличане на всички думи в текст, непосредствено след които се среща число.
- Извличане на всички думи в текст, след които се среща число, без да е фиксирано къде точно.

Упражнения

- Извличане на всички думи в текст.
- Извличане на всички думи в текст, непосредствено след които се среща число.
- Извличане на всички думи в текст, след които се среща число, без да е фиксирано къде точно.
- Извличане на всички думи в текст, които се срещат повече от веднъж.

Курсова задача I - примери

Задача:
В подаден текст всички дати във формат mm/dd/yyyy да се приведат във формат dd/mm/yyyy. Това да се направи само за датите, които със сигурност са във формат mm/dd/yyyy. Спорните дати (тези за които не е ясно в какъв формат са) да се изведат.
Вход: Низ.
Изход: Низ с извършените замени.
Изход (на екран): Поредица от спорните дати.

Задача:
Да се "инвертират" всички редове в таблици в даден html (т.е. последните колони стават първи, предпоследните - втори и т.н.). Счита се, че няма вложени таблици в html-а.
Вход: Низ в html формат.
Изход: Преобразуван низ в html формат.

Курсова задача I - примери (продължение)

Задача :
Да се изведат текстовете от всички листа на подаден xml, в които е дефиниран подаден атрибут с подадена стойност, и също така в текста се спомената тази стойност като отделна дума. При извеждането да се изведе и името на възела с атрибута.
Вход: Низ в xml формат, низ с име на атрибут и низ със стойност на атрибут.
Изход (на екран): Поредица от имената на листата и текстовете.

Задача:
В текст на английски език да се заменят всички изрази от вида “Xly Y” с “Y, in a X way,” или “Y, in an X way,”.
Вход: Низ.
Изход: Низ с извършените замени.

Курсова задача I - примери (продължение)

Задача:
От даден текст да се извлекат всички условия от вида "$X \geq n$" или "$X \leq m$", където n и m са числа, а X е подадена дума. Да се определят точните долна и горна граници за X.
Вход: Низ, който съдържа текста, и низ X.
Изход (на екран): Точните долна и горна граници на X.

Задача:
От даден текст да се извлекат всички съществителни, за които в същото изречение, в което се намират, има спомената сума в лева, евро или долари (€, EUR, $, USD, лв). Това, дали една дума е съществително, се разпознава чрез вече реализирана едноместна статична функция IsNoun.
Вход: Низ, който съдържа текста.
Вече реализирани методи: static bool IsNoun(string).
Изход (на екран): Поредица от всички съществителни.

Курсова задача I - примери (продължение)

Задача:
От даден текст да се извлекат всички синоними Y_i на подадена дума X и всички думи Z_i, за които условието "X е различно от Z_i" е налично в текста. Това, дали две думи са синоними, се разпознава чрез вече реализирана двуместна статична функция AreSynomyms. По извлечените данни да се формират два списъка - този от всички Y_i и този от всички Z_i.
Вход: Низ, който съдържа текста, и низ X.
Вече реализирани методи:
static bool AreSynomyms(string, string).
Изход (на екран): Двата списъка от Y_i и Z_i.

Курсова задача I - примери (продължение)

Задача:
От подаден код на C++/C#/Java да се извлекат имената на всички статични методи, чиято реализация не започва на същия ред, на който са декларирани.
Вход: Низ, който съдържа C++/C#/Java код.
Изход (на екран): Поредица от имената на намерените методи.

Задача:
От даден код на C#, който съдържа само публични методи, да се извлекат имената на всички методи, които са yield генератори и за метода не са дефинирани XML коментари.
Вход: Низ, който съдържа C# код.
Изход (на екран): Поредица от имената на намерените методи.