Национальный исследовательский университет «МИЭТ»

Институт интегральной электроники (группа ЭН-14М, каф. ПКИМС)

Компьютерные технологии в научных исследованиях

Лабораторная работа №2

Основы регулярных выражений в Tcl

Регулярные выражения

Регулярные выражения позволяют производить поиск подстрок в строке не на основании точного соответствия, а на основе некоторого шаблона.

```
regexp ?switches? template string ?matchVar? ?subMatchVar1? ...
```

```
regexp {[Y|y][E|e][S|s]} "Yes"
```

Шаблон — Строка, в которой ищется соответствие

Возвращаемое значение: 0 - если нет соответствия
1 – если соответствие есть

```
set a [regexp {[Y|y][E|e][S|s]} "String has yes in the middle"]
```

Что может входить в состав регулярного выражения?

В состав регулярного выражения могут входить:

1. литеральные символы,
2. наборы и классы символов,
3. итераторы,
4. операторы выбора,
5. вложенные шаблоны.

Литеральные символы

Простые символы – точное соответствие шаблону.

```
regexp {ab} "a"       → 0
regexp {ab} "b"       → 0
regexp {ab} "ab"      → 1
regexp {ab} "abba"    → 1
regexp {ab} "12ab34"  → 1
```

Универсальный заменитель – символ «.»

```
regexp {ab} "ab"      → 1
regexp {a.} "ar"      → 1
```

Наборы символов (1)

Конкретный выбор символа:

```
regexp {[Hh]ello} "A hello string"      → 1
regexp {[Hh]ello} "A Hello string"      → 1
regexp {[Hh]ello} "A hellO string"      → 0
regexp {[Hh]ello} "A hell string"       → 0
```

Допустимый диапазон символов:

```
regexp {[a-z]ello} "A mello string"     → 1
regexp {[a-z]ello} "A Hello string"     → 0
```

Наборы символов (2)

Объединение групп символов:

```
regexp {[a-z]ello} "A Hello string"        → 0
regexp {[a-zA-Z]ello} "A Hello string"     → 1
regexp {[a-z0-9]ello} "A 234ellostr"       → 1
```

Исключение отдельных символов или групп символов:

```
regexp {[^a-z]ello} "A Hello string"       → 1
regexp {[^a-z]ello} "A hello string"       → 0
regexp {[^a-z]ello} "A 234ellostr"         → 1
```

Классы символов (1)

Классом называется именованный набор символов

```
[:идентификатор:]
```

`[a-zA-Z]`	→	`[[:alpha:]]`
`[0-9]`	→	`[[:digit:]]`
`[ \b\n\r\t]`	→	`[[:space:]]`

```
regexp {[[:alpha:]][[:digit:]][[:alpha:]]} "a8a"    → 1
```

Классы символов (2)

Имя класса	Значение	Сокращение
`alnum`	Буквы верхнего и нижнего регистра, цифры	`\w`
`alpha`	Буквы верхнего и нижнего регистра	
`blank`	Пробелы и знаки табуляции	`\s`
`digit`	Цифры от 0 до 9	`\d`
`lower`	Буквы нижнего регистра	
`print`	То же, что и класс alnum	`\w`
`punct`	Знаки пунктуации	
`space`	Пробел, перевод строки, \n, \t и т.д.	`\s`
`upper`	Буквы верхнего регистра	
`xdigit`	Шестнадцатеричные цифры 0-9,a-f,A-F	

Итераторы

"*" - любое число вхождений предыдущего компонента

"+" – одно или более повторение

"?" – нулевое или единичное повторение

```
regexp ?switches? template string ?matchVar? ?subMatcVar1? ...
```

```
regexp {\w} "This is sample" var     → 1, var = "T"
regexp {\w*} "This is sample" var    → 1, var = "This"
regexp {\w+} "This is sample" var    → 1, var = "This"
```

Наиболее часто использующиеся ключи команды regexp

```
regexp ?switches? template string ?matchVar? ?subMatcVar1? ...
```

-nocase - делает сравнение без учёта регистра, формально переводя строку в нижний регистр

-expanded – позволяет добавлять комментарий к регулярному выражению в виде:

```
regexp -expanded {
  ^              # beginning of string
  [^:]+          # all characters to the first colon
  (?=            # begin positive lookahead
    .*\.com$     # for a trailing .com
  )              # end positive lookahead
} $x match
```

Якоря

Якорь "^" - признак начала строки

Якорь "$" - признак конца строки

```
regexp {^[0-9]+} "123 456 789" v      → 1, v = "123"
regexp {^\d+} "123 456 789" v
```

```
regexp {[0-9]+$} "123 456 789" v      → 1, v = "789"
regexp {\d+$} "123 456 789" v
```

Использование скобок

```
regexp template string ?matchVar? ?subMatcVar1? …
```

```
regexp {\w+(\d)\w+(\d)} "a1b2" v1 v2 v3
  v1 = "a1b2"
  v2 = "1"
  v3 = "2"
```

```
regexp {^module\s(\w+)} "module inverter (in, out);" a b
  a = "module inverter"
  b = "inverter"
```

Квантификаторы

```
set ret [regexp {(ab){3}} "pabababababf" a]
puts $ret
if {$ret} {
    puts stdout $a
}
```

```
set ret [regexp {(ab){3,}} "pabababababf" a]
puts $ret
if {$ret} {
    puts stdout $a
}
```

```
set ret [regexp {(ab){2,4}} "pabababababf" a]
puts $ret
if {$ret} {
    puts stdout $a
}
```

Домашнее задание

Скрипту передаётся в качестве аргументов командной строки перечень файлов. Нужно для каждого из них (при условии, что файлы существуют и являются файлами формата HSPICE - .sp или .cir) выполнить действие согласно варианту.

Для нечётных номеров вариантов:

С помощью регулярных выражений из входного нетлиста HSPICE вывести имена всех моделей и их уровень (level), для которых какой-либо параметр (выбрать) задаётся в единицах микрометров (с помощью U или u) (не 0.1u и не 12u).

Для чётных номеров вариантов:

С помощью регулярных выражений из входного файла моделей HSPICE вывести имена транзисторов и их моделей, для которых какой-либо параметр (выбрать) задаётся в единицах микрометров (с помощью U или u) (не 0.1u и не 12u).