Университет ИТМО

Кафедра ВТ

Системное программное обеспечение

Домашнее задание №2

Выполнил студент 2 курса
Группы P3211 Романов Олег
Преподаватель: Дергачев А.М

Санкт-Петербург
2016 год

Пусть переменная var имеет значение abc123. Определите, что возвращают следующие выражения - true или false

```
1. $var =~ /./            # true
2. $var =~ /[A-Z]*^/      # true
3. $var =~ /(\d)2(\1)/    # false
4. $var =~ /abc$/         # false
5. $var =~ /1234?/        # true
```

Пусть $var имеет значение abc123abc. Какое значение примет $var после следующих подстановок

```
1. $var =~ s/abc/def/;           # def123abc
2. $var =~ s/(?<=\d)[a-z]+/X/g;  # abc123X
3. $var =~ s/B/W/i;              # aWc123abc
4. $var =~ s/(.)\d.*\1/d/;       # abd
5. $var =~ s/(\d+)/$1*2/e;       # abc246abc
```

Чему соответствуют следующие шаблоны

```
1. /a|bc*/          # символ “a” или “b” и символ “c” 0 или больше раз
2. /[\d]{1,3}/      # от одной до трех десятичных цифр
3. /\bc[aou]t\b/    # слово cat или cot или cut
4. /(xy+z)\.\1/     # символ “x”, затем “y” один или больше раз, затем “z” и “.”, повтор выражения
   до точки
5. /^$/             # пустая строка
```

Написать шаблоны, соответствующие следующим примерам

1. не менее пяти маленьких латинских букв, либо буква "a", если она стоит в начале строки
   /[a-z]{5,}|^a/
2. цифра 1 или слово "one" (в любом регистре)
   /1|one/i
3. число, возможно, дробное (с десятичной точкой)
   /-?\d+\.?\d*/
4. любая буква, за которой следует гласная, повторяется еще раз (пример: "pop", "fifth", "daddy")
   /([a-zA-Z])[aeoiuy]\1/
5. хотя бы один "+"
   /\++/

Разобрать следующие команды

Ключи, используемые в командах:

- **e –** указывает, что далее следует Perl-выражение, которое нужно исполнить.
- **l –** все вводимые строки автоматически терминируются символом ‘\n’.
- **n –** оборачивает выражение в цикл вида:
  ```
  while (<>) {
        …
  }
  ```
- **p –** оборачивает выражение в цикл и печатает входные данные.
  ```
  while (<>) {
        …
  } continue {
        print or die “-p destination: $~\n”;
  }
  ```
- **m –** подключение модуля
- **C –** устанавливает режим работы с Unicode. С опцией SD устанавливает utf-8 для всех потоков.
- **0 –** установка разделителя записей файла ввода. Без параметров устанавливает \0, т.е. считывает весь код единой строкой.

1. perl -lne 'print if /((?<q>w)|a)(?(<q>)e|r)/'

Печатает каждую входную строчку, если она удовлетворяет условию: строка должна содержать в себе последовательность: “we” или “ar”.

2. perl -lpe '$p = qr/(\((?:[^()]++|(?-1))*+\))/; $_/=2 unless /x $p \+ y $p/x'

Передаваемая строка содержит две команды. Первая - это объявление перекомпилированное регулярное выражение, вторая строка, если входная строка не содержит выражения вида x$p+y$p , то делит входящее значение на два(если передано не число, то внутри будет лежать 0). Само регулярное выражение $p проверяет правильность скобочной последовательности (т.е. все скобки имеют закрытую пару). Используются (?:PATTERN) ? non-captured group(не имеет backtrace), и (?-PARNO) ? повтор группы под номером, кроме того используются два модификатора *+ и ++, которые аналогичны * и + за тем исключением, что не имеют backtrace. Итого, данная команда выводит на stdout либо результат деления числа на 2, выражение содержащие xR+yQ, где Q и R ? некоторые выражения с правильной скобочной последовательностью, к примеру C + x(W*t + radians(abs(ф))) + y(radians(ф)), и 0 в любом другом случае.
Проверка строки на чередование гласных с согласными, пропускате строки, содержащие что-либо помимо латинских букв.

3. perl -lne 'next if /[^a-z]/i;$v="aiueo";print if /^[^$v]?([$v][^$v])*[$v]?$/i'

Проверка строки на чередование гласных и согласных, пропускаются строки, содержащие что-либо помимо латинских букв.

4. perl -lne '@c=();for(split""){if(y/([{/)]}/){push@c,$_;next}if(/[])}]/){@c=(1),last if($_ ne pop@c);next}}print"F" if@c'

Если отформатировать передаваемую исполняемую строку, то получится подобный код:

```
@c=();
for (split"") {
if (y/([{/)]}/) {
push@c ,$_;
next
} if(/[])}]/) {
@c=(1), last if($_ ne pop@c);
next
}
}
print"F" if @c
```

Он делит каждую строку на отдельные символы, если символ является любой открывающей скобкой, то он заменяет его на закрывающего и помещает в конец массива @c, затем, если в строке встретился символ закрывающей строки, то сверит его с последним лежащим в массиве(pop с вершины стека) символом, если они равны, то продолжит выполнение, иначе поместит данный символ в массив, выйдет цикла for, если массив не пустой, то напишет F. Предназначение команды: проверка на правильность скобочной последовательности.

5. perl -lpe
'/^[^][{}()]*(([({[])((?:[^][{}()]*+|(?1)))*(??{$0=$2;$0=~y|([{|)]}|;"\\$0"}))/||($_=$.)'

Команда выводит номер строки во входном потоке или ej саму, если она удовлетворяет условию: Строка не должна начинаться с любой из скобок, затем содержать открытую скобку, после этого может содержать вcj что угодно кроме закрывающей скобки, затем содержит результат выполнения кода внутри конструкции (??). Результат данного выполнения скобка обратная скобке из второй capture-group. В итоге, данная команда проверяет опять правильность скобочной последовательности))0) Конструкция (?? code) так же как, как и (? code) выполняет код внутри фигурных скобочек во время применения этого регулярного выражения(ну если не стоят дополнительные ключевые слова для фаз компиляции вроде BEGIN, CHECK), но возвращаемое значение помещается внутрь внешнего, а не в переменную $^R

6. perl '-es!!),-#(-.?{<>-8#=..#<-*}>;*7-86)!;y!#()-?{}!\x20/`-v;<!;s++$_+ee'

Сначала вставляет в начало строки некоторую последовательность, затем заменяет символы в ней. Часть указана в ручную часть, из диапазона по ASCII-таблице. Затем производит снова замену, но уже с выполнием(модификатор ee) части заменителя($_). В итоге выполняет `cd /dev;sudo tee sdадля школоты >_< и слабо без sudo?

7. perl -pe 11..exit

Выполняет 11 итераций и завершает выполнение. Соль в range-operator .., который находясь в скалярном контексте работает иначе, чем в контексте списков. В скалярном контексте он работает как триггер: проверяется левый операнд, если false, то и всj выражение false, если true, то проверяется правая часть. Если операнд является константным выражением, это оператор будет истинен, только если равен номеру текущей строки в потоке входящих данных($.).

8. perl -pe 'print+(<>)[~9..-1]'

Печатает строки с 10 с конца до последней, затем самую первую. Работает так: формируется список из потока входящих строк, из него составляется список от -10 до -1 элементов. Унарный плюс используется для синтаксического разделения имени функции от скобочного выражения, которое будут интерпретированно, как список аргументов функции. Первая строка печатается в блоке continue.

9. perl -pe '$\=$_.$\}{'

Печатает строки из входного потока в обратном порядке. В переменную $\ записывается конкатенация $_ с предыдущим содержанием $\. Переменная $\ содержит разделитель выходных записей для функции print. Тонкий лайфхак: блок continue исполняется только после завершения цикла из-за наличия }{.

10. perl -CSD -Mutf8 -0pe 's@^* (.*?)[\r\n]+@<h3>$1</h3>@g' *

Команда берет в качестве аргумента все файлы текущей директории, затем находит в них строки с символом '*', пробелом, а затем неким текстом, этот некий текст помещается в capture-group, затем производится замена этой строки на заголовок html 3 уровня с содержимым capture-group.

11. perl -le '
```
        $LOVE=               AMOUR.
       true.cards.        ecstacy.crush
     .hon.promise.de    .votion.partners.
    tender.truelovers. treasure.affection.
   devotion.care.woo.baby.ardor.romancing.
   enthusiasm.fealty.fondness.turtledoves.
   lovers.sentiment.worship.sweetling.pure
   .attachment.flowers.roses.promise.poem;
    $LOVE=~ s/AMOUR/adore/g; @a=split(//,
     $LOVE); $o.= chr (ord($a[1])+6). chr
      (ord($a[3])+3). $a[16]. $a[5]. chr
       (32). $a[0]. $a[(26+2)]. $a[27].
         $a[5].$a[25]. $a[8].$a[3].chr
           (32).$a[29]. $a[8].$a[3].
             $a[62].chr(32).$a[62].
              $a[2].$a[38].$a[4].
                  $a[3].".";
                    print
                     $o'
```

Команда выводит “just another perl lover”.

- В переменную $LOVE записывается результат конкатенации разных слов.
- Слово AMOUR заменяется на adore.
- В массив @a записывается результат посимвольного разделения.
- Выводится строка составленная из символов массива @a, причем коды некоторых символов увеличиваются на определенные числа.