Лабораторная работа №1. Использование регулярных выражений в языке Perl.

16 сентября 2017 г.

Введение

Цель данной лабораторной работы — научиться работать с регулярными выражениями в языке Perl. Аналогичные регулярные выражения используются и в других языках программирования, в том числе в таких языках программирования высокого уровня, как Java или Python. В языке Perl регулярные выражения глубоко встроены в язык, что делает особенно эффективной обработку текстов с его использованием.

Для знакомства с языком рекомендуется использовать веб-сайт `http://perldoc.perl.org/`. Для знакомства с особенностями регулярных выражений в Perl рекомендуется использовать статью `http://perldoc.perl.org/perlretut.html`.

Форма отчетности: каждое задание сдается через проверяющую систему. В первом и втором разделе вы отправляете на проверку одну строку, которая интегрируется в существующую программу на Perl-е. В третьем задании вы пишете программу целиком.

Отчет о лабораторной работе представляет собой текстовый документ, содержащий ответы на задачи первого и второго раздела, принятые проверяющей системой. Письменный отчет по третьему разделу не требуется, достаточно самостоятельно написанных программ. В процессе защиты работы вы должны быть готовы объяснить их принцип действия и обосновать выбор тех или иных инструментов.

1 Фильтрация строк в файле

Задан текстовый файл, содержащий некоторое множество строк. Требуется вывести те из строк, которые удовлетворяют заданному кри-

терию. Будем задавать критерий в виде регулярного выражения. Рассмотрим, например, следующую программу.

```
while (<>) {
    print if /^(0|1)*(00|11)(0|1)*$/;
}
```

Регулярное выражение «`^(0|1)*(00|11)(0|1)*$`» допускает все слова, состоящие из нулей и единиц, в которых встречается два нуля подряд или две единицы подряд. Цикл читает из стандартного потока ввода строки и выводит те из них, которые подходят под регулярное выражение. Отметим, что ограничители «`^`» и «`$`» нужны, чтобы строка подходила под регулярное выражение от начала до конца. В противном случае Perl проверяет, есть ли в строке подстрока, подходящая под данное выражение. Например, следующая версия программы выводит все строки, содержащие подстроку «`pig`».

```
while (<>) {
    print if /pig/;
}
```

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`».

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`».

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`».

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`».

4. Строки, содержащие две буквы «z», между которыми ровно три символа. Пример строк, которые подходят: «zabcz», «zzz», «zzxzz». Пример строк, которые не подходят: «zz», «zxz», «zzxzxxz».

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

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».

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

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».

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 ».

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»,.

11. Строки, содержащие двоичную запись числа, кратного 3. Пример строк, которые подходят: «0», «10010». Пример строк, которые не подходят: «00101», «Not a number», «1 1», «0 0».

2 Преобразование строк в файле

Задан текстовый файл, содержащий некоторое множество строк. Требуется преобразовать каждую из этих строк в соответствии с заданным правилом и вывести результат. Используем следующую программу.

```
while (<>) {
    s/cat/dog/;
    print;
}
```

Эта программа заменяет первое вхождение подстроки «`cat`» в строке на подстроку «`dog`»: «`This cat is a nice cat`» → «`This dog is a nice cat`».

С модификатором «`g`» программа заменяет все вхождения подстроки «`cat`» в строке на подстроку «`dog`».

```
while (<>) {
    s/cat/dog/g;
    print;
}
```

«`This cat is a nice cat`» → «`This dog is a nice dog`».

В приведенных ниже задачах вам нужно написать регулярное выражение для описанного преобразования.

Обратите внимание, в большинстве заданий вам потребуются обратные ссылки.

1. Заменить все вхождения подстроки «human» на подстроку «computer». Примеры замен: «`I need to understand the human mind`» → «`I need to understand the computer mind`», «`humanity`» → «`computerity`».

2. Заменить все вхождения слова «human» на слово «computer». Запрещается использовать обратные ссылки. Указание: используйте «`\b`». Примеры замен: «`I need to understand the human mind`» → «`I need to understand the computer mind`», «`humanity`» → «`humanity`».

3. Заменить первое вхождение слова, состоящего только из букв «`a`» (регистр не важен) на слово «argh». Примеры замен: «`There'll be no more "Aaaaaaaaaaaaaaa"`» → «`There'll be no more "argh"`».

4. Поменять местами две первых слова в тексте. Примеры замен: «`this is a text`» → «`is this a text`», «`(This, ) is also a text`» → «`(is, ) This also a text`».

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

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

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

8. 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`».

9. Удалить символы после каждой открывающейся скобки до ближайшей закрывающейся. Примеры замен: «`(word) outside (1 open (2 open)`» → «`() outside ()`».

10. Будем называть *хорошей* строку, 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`».

3 Обработка файлов

В этом задании вам требуется написать целую программу, которая будет обрабатывать файл искомым образом. По возможности максимально используйте возможности регулярных выражений.

1. Будем называть строку пустой, если она состоит только из пробелов. Удалите начальные и конечные пустые строки во входном файле. Последовательности из двух или более пустых строк замените на одну полностью пустую строку.

Во всех остальных строках удалить ведущие и концевые пробелы, а последовательность из двух или более пробелов подряд заменить на один пробел.

Примеры

standard input
This is sample test   Remove extra empty lines and leading and   trailing spaces
standard output
This is sample test Remove extarempty lines and leading and trailing spaces

2. На вход подается HTML файл. Удалите все HTML теги вместе с их атрибутами. Оставшийся текст отформатируйте как в предыдущем задании.

Примеры

standard input
<html> <body> This is sample test   It contains some information   Quite stupid   </body> </html>
standard output
This is sample test It contains some information Quite stupid

3. На вход подается HTML файл. Найдите все ссылки вида «<a href="...">» в этом документе и выведите список сайтов, на документы которых он ссылается. Сайты следует выводить в алфавитном порядке, формат ссылки — см RFC 3986 (http://tools.ietf.org/html/rfc3986).

Примеры

standard input
<a href="http://neerc.ifmo.ru/school"> <a href="http://neerc.ifmo.ru"> <a href="ctddev.ifmo.ru:1328" >
standard output
ctddev.ifmo.ru neerc.ifmo.ru