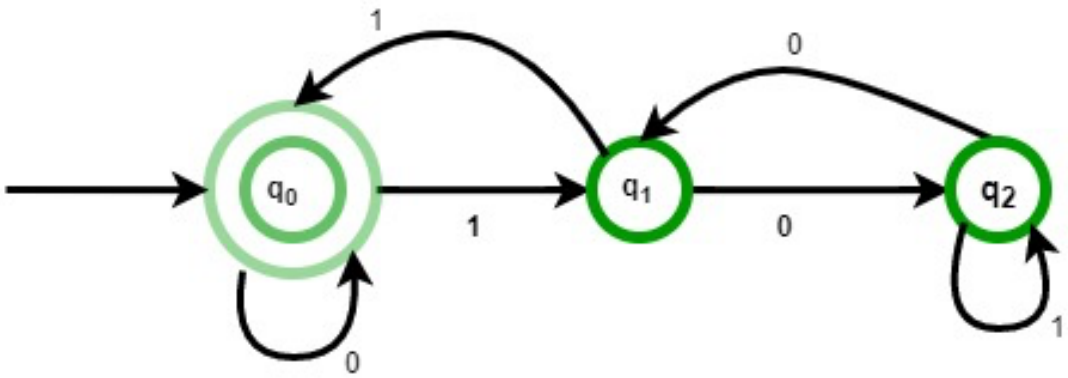
# Лабораторная работа №1. Использование регулярных выражений в языке Perl. Отчет.

## Задание 1.

1. Используем /cat.*cat/ Надо найти два вхождения строки cat и при проверке не учитывать символы между ними (отсюда использование .*).
2. Используем /(\W|^)cat(\W|$)/ Отделяем cat от всего вначале и конце через скобки до и после cat.
3. Используем /cat/i Слово мы ищем как подстроку (мы ничем его не экранировали в выражении) и добавляем /i как флаг игнорирования регистра.
4. Используем /z…z/ потому что три точки означают ровно три символа между z.
5. Используем /(x|y|z).{5,17}(x|y|z)/ В круглых скобках указываем какие буквы нас интересуют, в фигурных указываем количество символов между ними (для символа используем точку).
6. Используем /(\W|^)\d+(\W|$)/ Считаем минимум одну цифру и экранируем слева и справа от ненужных символов с помощью \W.
7. Используем /\\/ Экранируем \ с помощью \ .
8. Используем /\([^()]*\w+[^()]*\)/ По краям ставим скобки, внутри которых должно быть слово. Ищем само слово внутри скобок через \w+. Исключаем скобки внутри скобок с помощью [^()]*.
9. Используем /^(\S.*\S|\S)$/ Исключаем последовательность букв, окруженную пробелами и строки только из пробелов.
10. Используем /\b(\w+)\g1\b/ Внутри слова (благодаря паре \b) ищем последовательность символов, а затем повторяем ее с помощью бэкреференса \g1.
11. Используем /^(0*|1(01*0)*1)+$/ Если выпишем подряд числа, которые делятся на 3 в двоичной записи, то можно увидеть закономерность. 0, 011, 110, 1001, 1100, 1111, 10010 и тд. Составим ДКА.




Теперь по нему составим регулярное выражение. Произвольное количество нулей вначале для q0. Если мы из него выходим, то обязательно добавляем 1 вначале и 1 в конце (от/в q1). Переходы от/в q2 могут быть произвольное количество раз, отсюда *. 0 Чтобы вернуться из q2 в q1 обязателен (как и при q1 в q2). 1* есть переход q2 в q2.

## Задание 2.

1. Используем s/human/computer/g; Как в примере: заменяем все human на computer. Все просто.
2. Используем s/\bhuman\b/computer/g; Выделяем слова (благодаря паре \b) и заменяем на computer.
3. Используем s/\b[Aa]+\b/argh/; Выделяем слова (благодаря паре \b) в виде хотя бы одного A или a и заменяем на argh.
4. Используем s/\b(\w+)(\W*)(\w+)\b/$3$2$1/; Выделяем слова (благодаря паре \b) и находим первое и второе слово, убирая между ними ненужные нам символы. Затем меняем первое и второе слово, полученные в первой и третьей скобках регулярного выражения соответственно.
5. Используем s/\b(\w)(\w)/$2$1/g; Вначале каждого слова (благодаря единственному экземпляру \b) выделяем через \w первую и вторую букву. Затем меняем их местами, аналогично предыдущему заданию.
6. Используем s/(\w)\g1/$1/g; Ищем вхождения двух одинаковых букв подряд с помощью \w (это первая) и \g1 (а это вторая такая же) и заменяем найденную пару на одну букву, с помощью $1.
7. Используем s/(\w)(\g1+)/$1/g; Аналогично предыдущему заданию, но после первой буквы ищем одну и более таких же букв (через \g1+).
8. Используем s/\b(\d+)0\b/$1/g; В конце каждого числа (благодаря паре \b) отделяем 0, выделяя все старшие разряды в числе через \d+. Затем используем только эту часть, без нуля.
9. Используем s/\(.*?\)/()/g; Выделяем скобки, чтобы оставить только их. Внутри удаляем все, только не жадным образом, чтобы взять именно первую закрывающую скобку.
10. Используем s/(a.*?a){3}/bad/g; Выделяем ближайшие буквы a и символы между ними. Проверяем чтобы такая последовательность была три раза подряд. Заменяем такую подстроку на bad.

## Задание 3.

1. Код выглядит так (в круглых скобках в конце каждой строки написан комментарий):

```
my @array; (Объявим результирующий массив)
my $flag = 0;(Объявим флаг для контроля идущих пустых строк подряд)
while (<>) { (для всех строк)
    s/^\s*//; (Убираем все пробелы вначале строки)
    s/\s*$//; (Убираем все пробелы в конце строки)
    s/\s+/ /g; (Заменяем все последовательности в хотя один пробел
пробелов на один)
    if (/^$/) { (Если строка пустая)
        if ($flag == 0) { (И она первая )
            push @array, $_ (Добавляем в результат)
        }
        $flag = 1; (Запоминаем что идем в блоке пустых строк)
    } else { (Иначе, если строка не пустая)
        $flag = 0; (Обнуляем флаг, блок посты строк закончился)
        push @array, $_ (Добавляем в результат)
    }
}
while ($#array != -1 && @array[0] =~ /^$/) { (пока массив непустой и
первая строка пустая)
    shift(@array); (Убираем такую строку из начала массива)
}
while ($#array != -1 && @array[$#array] =~ /^$/) {(пока массив непустой и
последняя строка пустая)
    pop(@array); (Убираем такую строку из конца массива)
}
foreach my $line (@array) { (выводим все строки результата)
```

```
    print "$line\n";
}
```

2. Код выглядит так (комментарии оставлены только под новые требования для задания):

```
my @array;
my $flag = 0;
while (<>) {
    s/^\s*//;
    s/\s*$//;
    s/\s+/ /g;
    s/<[^>]+>//g; (ищем теги по ближайшим «<» и «>», убираем их и все
атрибуты внутри них)
    if (/^$/) {
        if ($flag == 0) {
            push @array, $_
        }
        $flag = 1;
    } else {
        $flag = 0;
        push @array, $_
    }
}
while ($#array != -1 && @array[0] =~ /^$/) {
    shift(@array);
}
while ($#array != -1 && @array[$#array] =~ /^$/) {
    pop(@array);
}
foreach my $line (@array) {
    print "$line\n";
}
```

3. Код выглядит так (в круглых скобках в конце каждой строки написан комментарий):

```
my @urls = (); (список ссылок)
my @sites = (); (результирующий список сайтов)
my $in = ""; (входные данные)
while(<>) {
    $in = $in.$_; (конкатенируем входные данные)
}
while ($in =~ /<\s*a.*?\bhref\s*=\s*"(\s*([^\"]*)\s*)".*?>/i) { (находим
первую в строке ссылку внутри HTML-тега)
    push(@urls, $1); (сохраняем ее)
    $in =~ s/<\s*a.*?\bhref\s*=\s*(\"\s*([^\"]*)\s*\").*?>//i; (убираем
из входных данных HTML-тег)
}

foreach (@urls) { (для каждой ссылки)
    /(?<scheme>([^\/:?#]+:)?\/\/)?(?<userinfo>(\w+(:\w+)?@))?(?<host>[^:
\/?#]+)(?<port>\:\d)?([:\/?#].*)?/; (разделяем имеющуюся ссылку на
различные части)
    $scheme = $+{scheme}; (схема обращения к ресурсу)
    $userinfo = $+{userinfo}; (логин и пароль пользователя)
    $host = $+{host}; (доменное имя хоста)
    $port = $+{port}; (порт хоста для подключения)
    if(!($scheme =~ /^\s*$/) or !($port =~ /^\s*$/)) { (если схема или
порт не пустые)
        push(@sites, $host); (запоминаем хост)
    }
}
```

```
my $prevURL = " "; (предыдущий хост)
foreach (sort(@sites)) { (выводим результирующий список отсортированно)
    if ($prevURL ne $_ && !($_ =~ /^\s*$/)) { (если хосты не совпадают и
не пустые)
        print $_; (выводим)
        print "\n";
    }
    $prevURL = $_; (обновляем переменную с предыдущим хостом)
}
```