Kilka istotnych informacji

Omawiana funkcja index(string, substring) może być użyta z 3 argumentami:

index(string, substring, początek)

Przez co można np. wyszukiwać otwarcie i zamknięcie tagu HTML, wyszukując najpierw otwarcie tagu, następnie używając tej pozycji jako początek do wyszukiwania zamknięcia tagu.

Używając funkcji split w przypadku gdy dzielimy za pomocą znaku \, Perl może potraktować ten znak, nawet wyescape'owany (\\), jako regex gdyż funkcja split może w tym miejscu przyjmować również regex. W takim wypadku lepiej użyć zamiast `'\\'` regexu `/\\/`.

Perl – wyrażenia regularne

Wyrażenia regularne działają w oparciu o najszybsze możliwe dopasowanie, stąd jeżeli jest możliwe jakiekolwiek dopasowanie określone wyrażeniami, jest ono używane. Do uzyskania dopasowań w wyrażeniach regularnych można wykorzystać następujące polecenia:

. - odpowiada pojedynczemu dowolnemu znakowi. W przypadku gdy chcemy dopasować kropkę, musimy ją wyescape'ować : \.
[abc] – dopasuj znak jeżeli wynosi on a, b lub c.
[0-9] – Dowolna cyfra. Zapis skrócony \d
[a-z] – Dowolna mała litera
[A-Z] - Dowolna wielka litera
[^...] - Znaki za wyjątkiem podanych dalej. Np. [^0a-z] dopasuje wszystkie znaki które nie są zerem lub małą literą.
* - 0 lub więcej wystąpień poprzedniego znaku. Np. 3[0-9]* oznacza dowolną ilość cyfr o ile rozpoczyna się cyfrą 3.
+ - Podobnie jak * z tym, że 1 lub więcej wystąpień.
? - 0 lub 1 wystąpień.
{min,max} – ilość wystąpień znaku w przedziale <min..max>. Przykładowo [0-9]{1,3}- dopasuje 1-, 12-, 122-
{min, } - minimalna ilość wystąpień znaku
{, max} – maksymalna ilość wystąpień znaku
{x} - dokładnie x wystąpień znaku

Proszę pamiętać, że ilości znaków opisane przez min/max ograniczają znaki pozostałe! Przykładowo:

```
my $str="aa123456ba123456789012345bc";
$str =~ s/[0-9]{6}/E/g;
```

da nam w wyniku:

```
aaEbaEE345bc
```

(wyrażenie)+ - powtarzalne wyrażenie gdziekolwiek w tekście. np. (ab)+c dopasuje w **ab**c, **abab**c, **abababc**, ale nie abaa czy acab.

^wyrażenie – wyrażenie na początku linii. Wyrażenie dopasowywane jest od początku wiersza.

wyrażenie$ - wyrażenie na końcu linii. Wyrażenie dopasowywane jest od końca wiersza.

Przykładowo: ^k[a-z]*c$ - spełnione będzie dla słów "koc", "koniec", ale ze względu na obecność znaku $ dla wyrażenia "kaczka" już nie.

Znaki sterujące (. + ? * ^ [ ) stosując jako fragment dopasowania należy poprzedzać znakiem \ podobnie jak escape'ujemy znaki specjalne (\n , \t).

Przykład: sprawdzenie czy wyrażenie spełnia kryteria dla kodu pocztowego (dwie cyfry, kreska – cyfra):

```
$z =~ m/[0-9]{2}-[0-9]{3}/
```

Jak już zostało wspomniane, wyrażenia regularne dopasowują od początku jak tylko jest to możliwe. Należy zauważyć, że takie wyrażenie zostanie spełnione zarówno dla np. 30-059 jak i 003-0000001 – bowiem zostanie tu znalezione dopasowanie 0**03-000**0001. Również np. a**00-000**BC byłby spełniony. Pełny regex dla kodu pocztowego uwzględniający jego długość wyglądałby więc np. tak:

```
$z =~ m/^[0-9]{2}-[0-9]{3}$/
```

Funkcje i procedury
Funkcję/procedurę możemy w kodzie zapisać w dowolnym miejscu. Po napisaniu funkcji można jej użyć w następującym po nim kodzie. W ten sposób często w istniejących skryptach zdarza się napotkać rekurencyjną funkcję wprowadzoną w środek kodu i zaraz potem wywołaną.
Funkcję zapisujemy poleceniem **sub** :

```
sub wyjscie {
      print ”Koniec skryptu”;
      exit;
}
```

funkcję możemy wywołać:

```
wyjscie();
```

albo po prostu

```
wyjscie;
```

Argumenty w funkcji pojawiają się automatycznie w tablicy @_ . Przykładowo:

```
sub maks {
        if ($_[0] > $_[1])
        {
             return $_[0];
        }
        return $_[1];
}
```

Powiedzmy, że argumentów jest więcej:

```
sub maks {
        my $maksimum=-65536;
        foreach my $a (@_)
        {
                if ($a>$maksimum)
                {
                        $maksimum=$a;
                }
        }
        return $maksimum;
}
```

Wykonać możemy taką funkcję przez np.:

```
my @k=(2, 3, 17, 8);
print maks(1, 2, 7, 6, 4, -1, @k, 10);
```

Proszę zauważyć nietypowy sposób przesyłania danych „przez tablicę”.

Najczęściej spotyka się w skryptach właśnie tak napisane funkcje. Istnieje w nowszych wersjach Perla możliwość włączenia **eksperymentalnego** wsparcia dla funkcji z ograniczeniami argumentów przez umieszczenie `use feature 'signatures';` w nagłówku i wtedy możemy:

```
sub maks (my $jedna, my $druga) {
 . . .
```

Polecenie **shift** zwraca pierwszy napotkany argument funkcji i usuwa go z tablicy argumentów – wszystkie inne „przesuwają się w lewo”. Polecenia tego można również użyć do parametrów dla skryptu.

Uruchamianie zewnętrznych programów:

```
system(”ps -u $uzytkownik”);
```

Zadania:

Przepisz funkcje maksimum tak, aby zamiast pętli foreach używała pętli while i funkcji shift.

Napisz program, który przeszuka dany plik tekstowy pod kątem numerów dowodów osobistych (3 wielkie litery i 6 cyfr) a następnie zastąpi wszystkie ich wystąpienia 9 znakami X zapisując wynik do drugiego pliku. Napisz krótki przykładowy plik z losowymi numerami i uruchom na nim program.