Perl – operacje na plikach
W dużym stopniu będziemy mieli do czynienia z plikami tekstowymi. Dostęp do plików realizowany jest przez tzw. uchwyty – zmienne, których sama wartość niewiele nas interesuje, lecz które wskazują na plik.

Wersje Perla - pewna lokalna uwaga.
Jak większość oprogramowania, na naszym serwerze jest kilka wersji Perla. Wersja domyślnie używana to 5.0, nie wspiera ona wielu elementów w tym standardowego od ok. 2000 roku dostępu do plików. Przy obsłudze plików należy użyć wersji 5.8, czyli:

```
#!/usr/bin/perl
```

zamieniamy na

```
#!/usr/local/bin/perl
```

Co umożliwi działanie wersji 5.8 (ca. 2004) a nie 5.0 (z 1999).

Otwieranie pliku:

```
open (my $FH, '<', "/etc/passwd") or die("Cannot open file!");
```

Zmienna $FH jest naszym uchwytem. Odczytajmy plik linia po linii:

```
while (my $linia = <$FH> ) {
      print $linia;
}
```

Oczywiście możemy tutaj użyć instrukcji foreach (`foreach my $linia (<$FH>)`), ale to utworzy nam tymczasową tablicę zawierającą wszystkie linie z pliku, co nie jest korzystne pamięciowo. W pętli while za każdym razem z uchwytu odczytywana jest jedna linia.

Zamiast '<' możemy otworzyć plik do zapisu ">" lub dopisać do pliku ">>". Po zakończeniu korzystania z pliku należy go zamknąć (`close $FH;`) aby wyczyścić bufor i zapisać zaległe dane.

Bardziej rozbudowany przykład:

```
open(my $FH, '<', "/etc/passwd") or die("cannot access file!");
open(my $OPH, ">", "test.txt") or die("cannot access savefile!");

while (my $linia = <$FH>) {
        if (index($linia,"is2014") == -1) {
                next;
        }
        my @Q=split(':',$linia);
        print "Znalazlem: ".$Q[2]." ".$Q[4]."\n";
        print $OPH $Q[2]." ".$Q[4]."\n";
}

close $FH;
close $OPH;
```

Użyte funkcje:
`index($string, $substring)` – zwraca położenie $substring w $string. -1 gdy go nie znajdzie. Użyteczne w wyszukiwaniu.
`split($delimiter, $lancuch)` – zwraca tablicę ciągów po podzieleniu łańcucha wejściowego.
Uwaga: Podejście `print split(':', $linia)[1]` nie uruchomi się ze względu na nieznany typ podczas parsowania.
`print $HANDLE string...` – drukuje ciąg tekstowy do pliku wskazanego przez $HANDLE.

Gdy już przeczytaliśmy…
Jeżeli przeczytaliśmy odpowiednie linijki i chcemy się cofnąć, możemy przenieść się na początek poleceniem `seek($FH, 0,0)`.

Jeszcze jedne użyteczna instrukcja do operacji na tekście: substr – fragment ciągu znaków.

```
my $str="Ala ma kota";
print substr($str,4,2);
```

Wydrukuje:

ma

Można również używać tej instrukcji do zamiany fragmentów:

```
my $str="Ala ma kota";
substr($str,4,2,"miala");
print $str;
```

"Ala miala kota".

Dodatkowo używając "index" można wykonywać zamiany znanych części zmiennej. Długość łańcucha tekstowego uzyskujemy używając funkcji length($string).

Wprowadzenie do wyrażeń regularnych – operator =~
Podstawowym narzędziem do modyfikowania łańcuchów tekstowych w Perlu są jednak rozbudowane mechanizmy wyrażeń regularnych. Powyższy przykład można przepisać:

```
my $str="Ala ma kota";
$str =~ s/ma/miala/;
print $str;
```

z takim samym wynikiem.

Zawsze wygląda do tak:

funkcja/regex/zamiennik/modyfikatory;

Najczęściej stosowane są funkcje:
- s – zamień (substitute)
- m – dopasuj (match) – np. if ($str =~ m/ma/) { …

Często stosowane modyfikatory:
- g – global – zamienia wszystkie wystąpienia, nie tylko pierwsze
- i – ignore case – ignoruje małe/wielkie litery.
- x – eXclude whitespace – wyklucza białe znaki chyba, że są escape'owane.

Modyfikatory można łączyć np. s/ma/miala/ig

Zadanie
Napisz skrypt, który wyświetli ilośćużytkowników nieużywających powłoki tcsh. Do sprawdzenia użyj wyrażeń regularnych.