PERL

Practical Extraction and Report Language

Pathologically Eclectic Rubbish Lister

Literatura

- Larry Wall, Tom Christiansen, Randal L. Schwartz, Stephen Potter,
  „PERL – programowanie”.
- Tom Christiansen, Nathan Torkington,
  „PERL – receptury”.

Internet

http://www.perl.com

http://www.activestate.com

Pierwszy program

```
#!/usr/bin/perl

$i=5;
while($i > 0) {
  print ”hello world\n”;
  $i--;
}
```

Dane

1. skalary

`$calar`

Dane

2. tablice

`@rray`

Dane

3. hasze

%hash

Składnia

```
# to jest komentarz
$i = 10; # to też jest komentarz
```

Składnia
Skalary i ich literały – dane numeryczne

```
$var = 123;          # liczba całkowita
$pi = 3.14159;       # liczba rzeczywista
$c = 3e8;            # notacja naukowa
$c = 300_000_000;    # notacja czytelna
$o = 015;            # ósemkowo 13
$x = 0xd             # szesnastkowo 15
```

Składnia
Skalary i ich literały – dane tekstowe

```
# bez interpolacji i interpretacji
$txt = ’ala ma kota’;

# z interpolacją i interpretacją
$txt = ”ala ma kota”;

# przechwycenie stdout
$txt = `pwd`;
```

Działanie interpolacji

```
$ona = ’Ala’;
$to = ’kota’;

$txt1 = ’$ona ma $to’;
# txt1: $ona ma $to

$txt2 = ”$ona ma $to”;
# txt2: Ala ma kota
```

Działanie interpretacji

```
$txt1 = 'linia\nlinia';
print $txt1;
```

```
linia\nlinia
```

Działanie interpretacji

```
$txt1 = "linia\nlinia";
print $txt1;
```

```
linia
linia
```

Tablice i ich literały

```
@arr = (”alpha”,”bravo”,”charlie”);

$arr[0] = ”alpha”;

$arr[1] = ”bravo”;

$arr[2] = ”charlie”;
```

Tablice i ich literały

```
$arr[0]    # element nr 0 jest skalarem

@arr[0]    # element nr 0 jest tablicą

$arr[-1]   # pierwszy element od końca

$#arr      # numer ostatniego elementu
```

Tablice i ich podstawienia

```
($e1,$e2,$e3) = @arr;

# $e1 ← ”alpha”;
# $e2 ← ”bravo”;
# $e3 ← ”charlie”;
```

Tablice i ich podstawienia

```
# lewa strona krótsza niż prawa
($e1,$e2) = @arr;

# pomijanie elementów
(undef,undef,$e1) = @arr;
```

Podstawienia tablic bez tablic

```
$one = 1;

$two = 2;

($one,$two) = ($two,$one);
```

Użycie tablicy jako skalara

```
$i = @arr;      # długość tablicy

$argc = @ARGV; # tablica standardowa

$cnt = scalar(@arr)
```

Kontekst wywołania funkcji

```
stat EXPR

Użycie w kontekście skalarnym:
$x = stat ”file”;

$x == 0 : plik nie jest dostępny
$x != 0 : plik jest dostępny
```

Kontekst wywołania funkcji

```
stat EXPR

Użycie w kontekście tablicowym:
@x = stat ”file”;

    0 dev device number of filesystem
    1 ino inode number
    2 mode file mode (type and permissions)
    3 nlink number of (hard) links to the file
    4 uid numeric user ID of file's owner
    5 gid numeric group ID of file's owner
    6 rdev the device identifier (special files only)
    7 size total size of file, in bytes
    8 atime last access time in seconds since the epoch
    9 mtime last modify time in seconds since the epoch
   10 ctime inode change time in seconds since the epoch
   11 blksize preferred block size for file system I/O
   12 blocks actual number of blocks allocated
```

Hasze i ich literały

```
%codes =
(”pl” => ”Poland”,
 ”uk” => ”United Kingdom”,
 ”de” => ”Germany”);

$codes{”pl”} = ”Poland”;
$codes{”uk”} = ”United Kingdom”;
$codes{”de”} = ”Germany”;
```

Usuwanie elementu hasza

```
delete $codes{”pl”};
```

Sprawdzanie istnienia elementu

```
#dobrze
if(defined($codes{”pl”})) { … }

#źle
if($codes{”pl”}) { … }
```

Hasz predeklarowany %ENV

```
$val = $ENV{"var"};
$ENV{"var"} = $val;

$ENV{"PATH"} .= ”:/home/user/bin";
```

Operatory arytmetyczne

```
$a = 2; $b = 3;

$c = $a + $b;  # 5
$c = $a - $b;  # -1
$c = $a * $b;  # 6
$c = $a / $b;  # 0.66666666.....
$c = $a ** $b; # 8
$c = $a % $b;  # 2
```

Operatory napisowe

```
$a = 'aaa'; $b = 'bbb';

$c = $a . $b;  # 'aaabbb'
$c = $a x 3;   # 'aaaaaaaaa'
```

Operatory napisowe

```
$a = 123; $b = 4;

$a .= $b;  # $a = $a . $b;
$a x= $b;  # $a = $a x $b;

etc...
etc...
etc...
```

Operator potrójny

```
$c = $ok ? $a : $b;
@c = $ok ? @c : @b;
$c = $ok ? @a : @b;

# uwaga! niespodzianka!
($ok ? $a : $b) = 100;
```

Operator przesuwania

```
$a = 2;

$b = $a >> 1; # 1
$b = $a << 1; # 4
```

Operatory bitowe

```
$a = 7; $b = 1;

$c = $a & $b;  #  1
$c = $a | $b;  #  7
$c = $a ^ $b;  #  6
```

Operatory post- i pre- ++ i --

```
$a = 1;
++$a; $a++;
--$a; $a--;

# zagadka
$a = 4;
$b = $a++ * ++$a; # $c?...
```

Operatory logiczne „short circuit”

```
$c = $a && $b; # $a ? $b : $a;

$c = $a || $b; # $a ? $a : $b;

$c = ! $a;   # $a ? 0 : 1;
```

Operatory logiczne nowego stylu

```
$c = $a and $b;  # $a ? $b : $a;
$c = $a or $b; # $a ? $a : $b;
$c = not $a;   # $a ? 0 : 1;

open F,”/file” or die ”bye\n”;
```

Operatory porównania

```
$c = $a OP $b;
```

relacja	liczby	napisy	wynik
równość	==	eq	prawda/fałsz
nierówność	!=	ne	prawda/fałsz
większe	>	gt	prawda/fałsz
większe równe	>=	ge	prawda/fałsz
mniejsze	<	lt	prawda/fałsz
mniejsze równe	<=	le	prawda/fałsz
porównanie	<=>	cmp	-1,0,1

Operatory plikowe (niektóre)

```
$a = '/home/user/file';
$c = OP $a;
```

operacja	nazwa	wynik
-e	**exists**	plik istnieje
-r	**readable**	plik da się czytać
-w	**writable**	plik da się pisać
-d	**directory**	plik jest katalogiem
-f	**file**	plik jest plikiem

Czym jest prawda?

- każdy napis z wyjątkiem "" i "0" ma wartość **PRAWDA**
- każda liczba z wyjątkiem wartości 0 ma wartość **PRAWDA**
- każda wartość niezdefiniowana ma wartość **FAŁSZ**

Instrukcja warunkowa (prosta)

```
if(wyrażenie){
      instrukcja;
}
```

```
if($a > 0) {
   $dodatnie++;
}
```

Instrukcja warunkowa (złożona)

```
if(wyrażenie){
      instrukcja;
} else {
      instrukcja;
}
```

```
if($a > 0) {
      $dodatnie++;
} else {
      $niedodatnie++;
}
```

Instrukcja warunkowa (kaskada)

```
if(wyrażenie){
  instrukcja;
} elsif (wyrażenie) {
  instrukcja;
} else {
  instrukcja;
}
```

Instrukcja warunkowa (odwrotna)

instrukcja if(wyrażenie);

```
$dodatnie++ if ($a > 0);
```

Instrukcja warunkowa (zaprzeczona)

```
unless(wyrażenie){
  instrukcja;
}
```

```
if(!(wyrażenie)) {
      instrukcja;
}
```

Instrukcja warunkowa (zaprzeczona)

instrukcja unless(wyrażenie);

```
print "równe" unless($a ne $b);
```

Pętla „while”

```
while(wyrażenie) {
    instrukcja;
}
```

```
while($t[$i]) {
   $i++
}
```

Pętla „while”

instrukcja while(wyrażenie);

```
$i++ while($t[$i]);
```

Pętla „until”

```
until(wyrażenie) {
    instrukcja;
}
```

```
until($t[$i]) {
     $i++
}
```

Pętla „for”

```
for(wyr1; wyr2; wyr3) {
    instrukcja;
}
```

```
wyr1;
while(wyr2) {
  instrukcja;
   wyr3;
}
```

Nieskończona pętla „for”

```
for(;;) {
    :
    :
}
```

Pętla „for”

Pętla „for” nie ma postaci krótkiej.

O_O

Pętla „foreach”

```
foreach $ZMIENNA(@LISTA) {
    ...
}
```

```
$suma = 0;
foreach $ten(@tablica) {
   $suma += $ten;
}
```

Pętla „foreach”

```
foreach $ten(@tablica) {
    $ten--;
}
```

Sterowanie pętlą

```
last ETYK;

next ETYK;

redo ETYK;
```