Základy PERLu snadno a rychle

Začínáme

Začneme tak, že si vytvoříme třeba soubor **hello.pl** , do souboru napíšeme:

```
#!/usr/bin/perl
print "Hello world\n";
```

Pak soubor nastavíme jako spustitelný:

```
$ chmod +x hello.pl
```

A teď si náš první skriptík vyzkoušíme

```
$ ./hello.pl
Hello world
$
```

Popis: První řádek **#!/usr/bin/perl** říká skriptu aby k jeho spuštění použil program **/usr/bin/perl**.
Dále je tu funkce **print** , která vypisuje data na obrazovku.

Skalární proměnné

Skalární proměnné jsou proměnné, které mohou obsahovat různé typy dat.

Např.:

Číselné: 1, 2, 3, 1.32 , 5.86666 ...
Znakové: A , B , c, d, AHOJ, Dobrý den...

Deklarace skalární proměnné:

```
$promenna = "Abcd";
$a = 10;
$b = 12.26689;
   .
   .
   .
```

Poznámka: Skalární proměnné **nesmí** začínat číslicí, třeba **$5 = 1;** nebo **$49 = "abc";**
Ale **můžou** začínat třeba **$a45 = 1;** nebo **$b2 = "abcdef";**

Práce se skalárními proměnnými

Tento skript vypíše součet proměnných **$a** a **$b**.

```
#!/usr/bin/perl

$a = 1;
$b = 3;
$c = $a + $b;

print "$c\n";
```

Tento skript dělá úplně to samé jako předchozí skript, akorát je jinak zapsaný.
Znak **'.'** se používá pro spojování řetězců.

```
#!/usr/bin/perl

$a = 1;
$b = 3;
print $a + $b . "\n";
```

Bez použití proměnných by to taky šlo zapsat takhle:

```
#!/usr/bin/perl

print 1 + 3 . "\n";
```

Spojování a práce se řetězci

V tomto názorném skriptu si deklarujeme 3 proměnne třeba **$a**, **$b** a **$c** do proměnné **$a** uložíme **"Dobrý "** a do **$b** uložíme **"den."** a tyto proměnné spojíme do proměnné **$c** a pak proměnnou **$c** vypíšeme na obrazovku.

```
#!/usr/bin/perl

$a = "Dobrý ";
$b = "den.";
$c = $a . $b;

print "$c\n";
```

Taky by to šlo takhle:

```
#!/usr/bin/perl

$a = "Dobrý ";
$b = "den.";

print "$a$b\n";
```

Pole

Pole se identifikuje znakem **"@"**

Deklarace pole:

```
@pole = ("a", "h", "o", "j");
@a = (100, 5, 256, 33);
    .
    .
    .
```

Výpis obsahu pole:

```
print "@pole\n";
```

Poznámka: Výše uvedeý výpis vypíše obsah pole oddělený mezerami.

Když chceme vypsat určitý prvek pole, tak to provedeme pomocí zápisu:
$název_pole[prvek_pole]

```
#!/usr/bin/perl

@pole = ("1.prvek", "2.prvek", "3.prvek");

print "$pole[0] , $pole[1] , $pole[2]\n";
```

Funkce push() a pop()

Funkce **push()** slouží k přidávání prvků do pole.

Zápis: **push(@název_pole, obsah_prvku);**

```
#!/usr/bin/perl

@pole = ("Dobrý");

print "@pole\n";

push(@pole, "den");

print "@pole\n";
```

Funkce **pop()** slouží k odebrání posledního prvku pole a vrací jeho hodnotu.

Zápis: **pop(@název_pole);**

```
#!/usr/bin/perl

@pole = ("Dobrý", "den.");

print "@pole\n";

$posledni_prvek = pop(@pole);

print "@pole\n";

print "Poslední prvek pole je: $posledni_prvek\n";
```

If a else

If a **else** se používá k porovnávání proměnných, čísel, řetězců...
Vrací buď **TRUE**(pravda) nebo **FALSE**(nepravda)

Zápis:
if ($proměnná [operátor] $proměnná)
{
příkazy..;
.
.
.
}
else
{
příkazy..;
.
.
.
}

Základní operátory:
== (rovná se)
!= (nerovná se)
> (větší než)
< (menší než)
>= (větší nebo rovno)
<= (menší nebo rovno)

Tento skriptík nám bude porovnávat obsahy proměnných **$a**, **$b**, **$c**

```
#!/usr/bin/perl

$a = 1;
$b = 0;
$c = 1;
```

```
if ($a == $b)
{
  print "$a == $b\n";
}
else
{
  print "$a != $b\n";
}

if ($a == $c)
{
  print "$a == $c\n";
}
else
{
  print "$a != $c\n";
}

if ($c == $b)
{
  print "$c == $b\n";
}
else
{
  print "$c != $b\n";
}
```

Cykly while() a foreach()

Cyklus **while** opakuje příkazy pokud zadaná podmínka platí.

Zápis:
while(podmínka)
{
příkazy..;
.
.
.
}

Tento skript nám napočítá do deseti :)

```
#!/usr/bin/perl

$cislo = 1;

while ($cislo <= 10)
{
  print "$cislo\n";
  $cislo++;
}
```

Poznámka:
$cislo++;** je ekvivalentní zápisu **$cislo = $cislo + 1;
$cislo--;** je ekvivalentní zápisu **$cislo = $cislo - 1;

Cyklus **foreach()** se používá k práci s polem, postupně načítá první až poslední prvek pole a ukládá ho do zvolené proměnné.

Zápis:
foreach $proměnná (@pole)
{
příkazy..;
.
.
.

}

Následujicí skript nám z pole **@cisla** vypíše čísla větší než 5

```
#!/usr/bin/perl

@cisla = (1, 50, 13, 2, 6, 4, 3, 0, 9, 8, 7);

foreach $promenna (@cisla)
{
  if ($promenna > 5)
  {
    print "$promenna\n";
  }
}
```

Tento script nám ty samé čísla ještě roztřídí na menší a větší než 5 s pomocí funkce **push()**

```
#!/usr/bin/perl

@cisla = (1, 50, 13, 2, 6, 4, 3, 0, 9, 8, 7);

foreach $promenna (@cisla)
{
  if ($promenna > 5)
  {
    push(@vetsi, $promenna);
  }
  else
  {
    push(@mensi, $promenna);
  }

}

print "Cisla vetsi nez 5: @vetsi\n";
print "Cisla mensi nez 5: @mensi\n";
```

Práce se soubory, vstupem a funkce chop()

Soubory se otevírají pomocí funkce **open()** a zavírají pomocí **close()**
Každému souboru musí být přiřazen indentifikátor.

Zápis:
open(indentifikátor, "cesta/soubor"); [Otevření pro čtení]
open(indentifikátor, ">cesta/soubor"); [Otevření pro zápis]
open(indentifikátor, ">>cesta/soubor"); [Otevření pro zápis na konec souboru]

V tomto skriptu do souboru **text-file** uložíme text, pak ho ze souboru vypíšeme, na konec souboru připíšeme další text a pak obět soubor vypíšeme

```
#!/usr/bin/perl

open(F, ">text-file");
print F "Tohle je text na prvni radce souboru\n";
close(F);

open(F, "text-file");
print <F>;
close(F);

open(F, ">>text-file");
print F "Tohle je text pridany na konec souboru\n";
close(F);

open(F, "text-file");
print <F>;
close(F);
```

Vstup má konstantní identifikátor **STDIN**
Tento skriptík načte z klávesnice jeden řádek ukončený klávesou ENTER

```
#!/usr/bin/perl

print "Napis neco:";
$vstup = <STDIN>;
print "\n\nNapsal jsi: $vstup";
```

Funkce **chop()** odstraní poslední znak z proměnné a vrátí jeho hodnotu, používá se hlavně když chceme něco načíst z klávesnice a pak porovnat s nějakou jinou proměnnou. Protože při načítání z klávesnice se ukládá i znak **"\n"**, tak je vhodné použít tuto funkci k jeho odebrání.

V tomhle závěrečném skriptu si ukážeme použití **vstupu**, cyklu **while**, funkce **chop()** aj.

```
#!/usr/bin/perl

$heslo = 123654;
$pokus = 1;

while($heslo != $vstup)
{
  print "($pokus. pokus) Zadej heslo:";
  $vstup = <STDIN>;
  chop($vstup);

  if ($vstup != $heslo)
  {
    print "\nSpatne heslo, zkus to znovu\n\n";
  }
  $pokus++;
}

print "\n\nHeslo je OK\n";
```

Toto je pro dnešek vše, přeji vám hezký den.

autor: Antonín Brettšnajdr