PB173 Perl

02 Zoznamy, polia

Roman Lacko <xlacko1@fi.muni.cz>

Obsah

Zoznamy

- Hodnoty v zátvorkách oddelené `,`

```
();
(1, 2, 3);
("red", "green", "blue");
```

- Hodnoty nemusia mať rovnaký „typ“

```
(42, "Perl", $x, \$y, sub ($n) { ... });
```

Operácie nad zoznamom

Zoznamy je možné indexovať:

```
my $x = (1, 2, 3)[0];
my $last = (1, 2, 3)[-1];
```

Je možné z nich robiť výrezy (*slices*):

```
my ($a, $b) = (1..10)[1, 2];
```

Dajú sa konštruovať opakovacím operátorom:

```
(VALUE) x $count;
($a, $b) x $count;
```

Odbočka: Úvodzovkovité operátory

`q{STRING}`
Ekvivalent `'STRING'`

`qq{STRING}`
Ekvivalent `"STRING"` vrátane interpolácie

`qw{STRING}`
Vráti zoznam slov zo `STRING` rozdelených podľa medzery.

Okrem `X{}` je možné použiť aj `X( )`, `X[]`, `X<>`, `X''`, `X//`, ...

```
qw(Hello World);
('Hello', 'World');
```

Pole

Premenná začínajúca značkou @ je *pole*:

```
my $scalar;
my @array;
```

Pole môžeme inicializovať zoznamom:

ex01t-arrays.pl

```
my @primes = (2, 3, 5, 7, 11);
my @langs = qw(Perl C C++ Java);
```

Interpolácia poľa

Celé pole môžeme vypísať interpoláciou:

```
say "@names";
```

Pri interpolácii sa medzi prvky vkladá hodnota špeciálnej premennej `$"`.
Porovnajte výstup s nasledujúcim kódom:

ex01-arrays.pl

```
$" = ':';
say "@names";
```

Bude výpis iný pre `say @names;`?

Prístup k prvkom

K prvkom poľa pristúpime operátorom `[INDEX]`. Prvý prvok má index `0`.

Sigil na začiatku premennej nehovorí len, akého „typu“ je premenná,
ale aj aký „typ“ hodnoty očakávame.

`ex02t-array-access.pl`

```
my $sheep = 5;
my @sheep = qw(Mary Jane Elisa Shrek);

say "Scalar: $sheep, Array: @sheep";
say "Value from array: $sheep[1]";
```

Premenné `$langs` a `@langs` nie sú ten istý objekt!

Fun fact: Premenná `$[` mohla do verzie `v5.30` meniť index prvého prvku.

Výrezy

Výrezy z poľa môžeme robiť podobne ako pri zozname

```
my @herd = @sheep[1..3];
```

- Ako vyrobíme výrez veľkosti 1?
- Čo ak je veľkosť výrezu väčšia než pole?

Vyskúšajte `ex03p-slice.pl`.

Zoznam vs Pole

V Perli:

- *Zoznam* nie je skutočná dátová štruktúra, nemôže existovať v premennej.
- *Pole* je dátová štruktúra, ktorá žije v premennej typu @X.

Pole môžeme obvykle použiť tam, kde sa očakáva zoznam, napríklad:

- Výrezy

```
@array[1, 2, 3]
@array[@indices]
```

- Volania funkcií

```
function(1, 2, 3);
function(@params);
```

Zoznam vs Pole

Niekedy sa však chovajú veľmi odlišne. Porovnajte:

ex04-context.pl

```
my $lv = (5, 6, 7, 8);

my @array = (5, 6, 7, 8);
my $sv = @array;
```

Prečo sú `$lv` a `$sv` odlišné? Čo ich hodnota znamená?

Kontext

Perl rozlišuje *kontext*, v ktorom sa výraz nachádza:

prázdny (*void*) kontext
: Hodnota výrazu sa nepoužije ani neuloží.

skalárny kontext
: Hodnota výrazu sa použije ako skalár.

zoznamový kontext
: Hodnota výrazu sa použije ako zoznam.

Zoznam v rôznych kontextoch

Zoznam v skalárnom (a prázdnom) kontexte vyhodnotí všetky prvky a vráti posledný:

```
my $x = (1, 2, 3);
my $x = 1, 2, 3;
```

Zoznam v zoznamovom kontexte priradzuje svoje prvky na ľavú stranu:

```
my ($x, $y) = (1, 2, 3);
my @array = (1, 2, 3);
```

Pole v rôznych kontextoch

Pole v skalárnom kontexte vráti svoju veľkosť:

```
my $size = @array;
```

Pole v zoznamovom kontexte sa správa ako zoznam:

```
my @new_array = ('1', '2', @l1, @l2);
```

```
my @args = ("Hello, %s\n", "Hello", $name);
printf @args;
```

Vynútenie kontextu

Skalárny kontext je možné vynútiť operátorom `scalar`:

```
scalar EXPRESSION;
```

Pre zoznamový kontext špeciálny operátor neexistuje,
obvykle stačí použiť zátvory (`(EXPRESSION)`).

Vyskúšajte `ex05p-size.pl`

Operácie s poľom

`$#array` vráti index posledného prkvu alebo `-1`.

```
for my $index (0 .. $#array) {
    say $array[$index];
}
```

Ak nás index nezaujíma, môžeme iterovať priamo hodnoty:

```
foreach my $value (@array) {
    say $value;
}
```

`$value` je v tomto cykle *alias* pre hodnotu v poli.
Zmenou `$value` teda meníme pôvodné pole.

Operácie s poľom

Vkladanie a výber prvkov z konca poľa:

```
push @array, LIST...;
pop @array;
```

Vkladanie a výber prvkov zo začiatku poľa:

```
unshift @array, LIST...;
shift @array;
```

Operácie s poľom

Pole môže obsahovať akékoľvek skalárne hodnoty, vrátane `undef`.

```
defined $array[N];
```

- Otestuje, že hodnota v poli je definovaná.

Prístup mimo pole sa vždy vyhodnotí na `undef`.

```
exists $array[N];
```

- Otestuje, že pole má hodnotu na indexe `N`.

Operácie so poľom

```
splice @array, $offset, [$length, [LIST...]];
```

- Z poľa `@array` zmaže prvky od pozície `$offset`.
  Ak `$offset < 0`, tak sa berie `-$offset` od konca.
- Pre parameter `$length` zmaže len zadaný počet prvkov.
  Ak `$length < 0`, maže do konca okrem posledných `-$length` prvkov.
- Ak je zadaný aj zoznam, tak zmazané prvky sa nahradia prvkami zoznamu.
- Vráti všetky zmazané prvky alebo posledný podľa kontextu.

Vyskúšajte si `splice` v príklade `ex07p-splice.pl`.

Operácie s poľom reťazcov

```
split STRING, EXPR, [LIMIT]
```

- Rozdelí hodnotu výrazu `EXPR` podľa znaku alebo reťazca v `STRING`.
- Vráti zoznam všetkých hodnôt, alebo len prvých `LIMIT` z nich

`split` sa častejšie používa s regulárnymi výrazmi.

```
join STRING, LIST...;
```

- Spojí hodnoty v zozname reťazcom `STRING` a vráti nový reťazec.