PB173 Perl

03 Asociatívne polia, referencie

Roman Lacko xlacko1@fi.muni.cz

2023-02-27

Obsah

Asociatívne polia

Asociatívne polia

Pole je premenná, ktorá drží **zoznam**. Má sigil ‹`@`›:

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

Asociatívne pole

Asociatívne pole (hash) je *neusporiadané* pole dvojíc (kľúč, hodnota).
Kľúče sú vždy reťazce. Má sigil `%`.

```
my $scalar;
my @array;
my %hash;
```

Asociatívne pole: Iniciaizácia

Hash inicializujeme zoznamom.
Na rozdiel od poľa musí mať párny[1] počet prvkov:

- *párne* pozície sú kľúče, všetky sa interpretujú ako reťazce,
- *nepárne* pozície sú hodnoty.

```
my %languages = (
    'cs', 'Czech',
    'sk', 'Slovak',
    'uk', 'English (traditional)',
    'us', 'English (simplified)',
);
```

[1] sudý

Asociatívne pole: Operátor ‹=›

Operátor ‹=› sa správa ako ‹ , ›, ale svoj ľavý operand zmení na reťazec.
Nasledujúce zoznamy sú ekvivalentné:

```
("foo", "bar")
(foo => "bar")
```

Asociatívne pole: Operátor ‹=›

Typické použitie ‹=› je pri inicializácii hashu:

```
my %languages = (
    sk => 'Slovak',
    cs => 'Czech',
    uk => 'English (traditional)',
    us => 'English (simplified)',
);
```

Čo ak chceme ako kľúč použiť návratovú hodnotu funkcie?

Asociatívne pole: Duplikáty

Pri **inicializácii** zoznamom s násobnými kľúčmi sa ponechá posledné priradenie:

```
my %wizard = (
    name => 'Mithrandir',
    name => 'Olórin',
    name => 'Tharkûn',
    name => 'Gandalf',
);

say "@{[ %wizard ]}";      # name Gandalf
```

Asociatívne pole: Duplikáty

Pozor však na hodnoty, ktoré nie sú reťazce:

```
my %zeroes = (
    0 => 'integer',
    0.0 => 'float',
);

say "@{[ %zeroes ]}"    # 0 float
```

Čo sa stalo?

Asociatívne pole: Poradie prvkov

Pri **iterácii** vracia striedavo kľúče a k nim naviazané hodnoty.
Poradie kľúčov však nie je v žiadnom definovanom poradí!

```
my %nums = (one => 1, two => 2);
say "@{[ %nums ]}";

# Might say one of these two:
#   one 1 two 2
#   two 2 one 1
```

Hash (na rozdiel od poľa) nevie reťazcovú interpoláciu.
(Netýka sa prvkov, `say "$hash{key}"` funguje)

Asociatívne pole: Indexácia

K prvkom hashu pristúpime operátorom ‹ {} ›.
Podobne ako pri poli, aj tu *sigil* znamená „typ“ hodnoty, ktorú očakávame:

```
say $languages{'en'};
```

Ak je index len jedno slovo, môžeme úvodzovky vynechať.

```
say $languages{en};
```

Asociatívne pole: Výrezy

Pre hash môžeme vytvoriť dva typy výrezov

- Výrez hodnôt je **zoznam** hodnôt pre uvedené kľúče:

```
my @values = @languages{'sk', 'cs'};
# @values = ('slovak', 'czech')
```

- Výrez hashu je **zoznam** vybraných kľúčov a ich hodnôt:

```
my @values = %hash{'cs', 'uk'};
# @values = ('cs', 'czech', 'uk', 'proper english')
```

Asociatívne pole: Výrezy

Pre pole môžeme teraz vyrobiť ďalší typ výrezu s hashom:

- Výrez hodnôt je **zoznam** hodnôt na vybraných indexoch (videli sme minule)

```
my @values = @array[1, 10, 100];
# @values = ('ab', 'ak', 'dw');
```

- Výrez s indexami je **zoznam** vybraných indexov a hodnôt na nich:

```
my @values = %array[1, 10, 100];
# @values = (1, 'ab', 10, 'ak', 100, 'dw');
```

Asociatívne pole: Výrezy

Aj keď je poradie kľúčov v hashi nedeterministické,
výrez je vždy **zoznam** prvkov v poradí, ktoré sme žiadali:

```
my ($login, $name, $password) = @user{qw{login name passwd}};
```

Do výrezu je tiež možné priradzovať hodnoty:

```
@user{'name', 'surname'} = split " ", $line;
```

Asociatívne pole: Výrezy

Pre zhrnutie:

- ‹`@`› alebo ‹`%`› používame podľa toho, čo má byť **výsledok**,
- ‹`[]`› alebo ‹`{}`› používame podľa toho, z **čoho** vyberáme hodnoty

```
@x[INDICES];    # @ → we want array   [] → ‹x› is array
%x[INDICES];    # % → we want hash    [] → ‹x› is array

@x{KEYS};       # @ → we want array   {} → ‹x› is hash
%x{KEYS};       # % → we want hash    {} → ‹x› is hash
```

Asociatívne pole: Výrezy

Koľko rôznych premenných používa tento výraz?

```
$x{$x[$x]}
```

- `$x` skalárna premenná `$x`
- `$x[…]` pole `@x`
- `$x{…}` hash `%x`

Asociatívne pole: Kontext

Hash v skalárnom kontexte vráti počet kľúčov.

```
say scalar %languages;      # 4
```

Pred ‹v5.25› bol výsledkom reťazec s informáciou o zaplnení hash tabuliek

Hash v zoznamovom kontexte vráti zoznam, kde sa striedajú kľúče a k nim priradené hodnoty.

Poradie týchto hodnôt nie je garantované. Nespoliehajte sa naňho.

Asociatívne pole: Operácie

```
keys HASH
values HASH
```

- Vráti zoznam kľúčov, resp. hodnôt.
- Od ‹v5.12› funguje aj na poli, vráti zoznam indexov resp. hodnôt.

Kľúče ani hodnoty hashu nie sú v žiadnom vopred určenom poradí.
Platí však ‹$hash{(keys %hash)[INDEX]}› ≈ ‹(values %hash)[INDEX]›.

Ak chcete kľúče zoradiť, použite ‹sort keys HASH›.
‹sort› uvidíme podrobnejšie neskôr.

Asociatívne pole: Ďalšie operácie

```
exists EXPRESSION
defined EXPRESSION
```

Podobne ako pre pole overí existenciu kľúča, resp. či je definovaná hodnota.

```
$hash{u} = undef;

say exists $hash{u};      # This exists …
say defined $hash{u};     # … but is not defined.
```

Asociatívne pole: Ďalšie operácie

```
delete EXPRESSION
```

Zmaže hodnotu a kľúč z hashu. Môžeme uviesť aj výrez.

```
delete $languages{cs};
delete @languages{qw{cs sk}};
delete %languages{qw{cs sk}};
```

Vráti zmazanú hodnotu alebo zmazané výrezy.

> ‹delete› **na poli**
> Tento operátor je možné použiť aj na pole a jeho výrezy.
> Nie je to však odporúčané; použite ‹splice›.

Asociatívne pole: Iterácia

Ako iterovať cez kľúče a hodnoty?

```
foreach my $key (keys %languages) {
    say "$key: $languages{$key}";
}
```

Asociatívne pole: Iterácia

```
each HASH
each ARRAY
```

Pri každom volaní vráti ďalšiu dvojicu `(key, value)`, po poslednej `()`.
Iterátor je viazaný na daný kontajner.

```
while (my ($key, $value) = each %hash) {
    say "$key: $value";
}
```

Čo sa stane, ak vnoríme `each` na tom istom hashi?

Asociatívne pole: Iterácia

Iterácia s kľúčom
Táto vlastnosť je experimentálna od `v5.36`

```
use experimental 'for_list';

foreach my ($key, $value) (%languages) {
    say "$key: $value";
}
```

Toto rozšírenie môžete používať v úlohách

Referencie

Referencie

Hodnotou skalárnej premennej môže byť odkaz (*referencia*) na inú hodnotu.

Referencie vytvárame operátorom `\`.

```
my $a = 42;
my $ref_a = \$a;
my $ref_number = \32;
my $ref_string = \"Perl";
my $ref_undef = \undef;
```

Odkazovanou hodnotou nemusia byť len skaláry.

```
my $array_ref = \@array;
my $hash_ref = \%hash;
```

Circumfix notácia

```
# SIGIL { EXPR }     # Or without { } for simple expressions.
${$ref}, $$ref       # scalar
@{$ref}, @$ref       # array
$#{$ref}, $#$ref     # array (last index)
%{$ref}, %$ref       # hash
```

Circumfix notácia

Pri výbere hodnoty z referencie na pole alebo hash meníme sigil:

```
@$array_ref
$$array_ref[INDEX]
```

```
%$hash_ref
$$hash_ref{KEY}
```

Podobne pre výrezy:

```
@$array_ref[INDICES…]
@$hash_ref{KEYS…}
%$hash_ref{KEYS…}
```

Postfix notácia (od ‹v5.20›)

```
# EXPR -> SIGIL *
$ref->$*            # scalar
$ref->@*            # array
$ref->$#*           # array (last index)
$ref->%*            # hash
```

‹->› je medzi zátvorkami nepovinná. Nasledujúce výrazy sú ekvivalentné:

```
$canvas->{alpha}->[$x]->[$y]->{pixel} = $pixel;
$canvas->{alpha}[$x][$y]{pixel} = $pixel;
```

Postfix notácia (od ‹v5.20›)

Pri výbere hodnoty z referencie na pole alebo hash stačia zátvorky:

```
$array_ref->@*
$array_ref->[INDEX]
```

```
$hash_ref->%*
$hash_ref->{KEY}
```

Sigils však potrebujeme pre výrezy:

```
$array_ref->@[INDICES…]
$hash_ref->@{KEYS…}
$hash_ref->%{KEYS…}
```

Ktorú notáciou používať?

Postfix notácia (`$ref->Σ*`) môže vyzerať na prvý pohľad zložitejšie, než circumfix (`Σ$ref`). Lepšie sa však číta pri hlbokých štruktúrach.

Z poľa kníh uložených ako hashe chceme autorov prvej knihy

```
@{$${$$books[0]}{authors}}
```

```
$books->[0]->{authors}->@*
```

Používajte ten spôsob, ktorý sa vám páči viac, buďte však **konzistentní**.

Ak chcete používať staršiu (circumfix) notáciu,
dávajte si pozor na zátvorky.

Aký je rozdiel medzi týmito výrazmi?

```
say $$$ref[0];
say ${$$ref}[0];
say ${$$ref[0]};
say $$ref[0][0];
```

Referencie: Interpolácia

Použitie referencie v reťazcovom kontext vráti nejakú internú reprezentáciu:

```
my $array_ref = \@languages;
say $array_ref;                    # ARRAY(0x55d612b044b8)
```

Nepoužívajte referencie ako kľúče v hashoch!

Referencie: Priradenie

Obyčajné priradenie do premennej referenciu zruší:

```
$a = 42;
$b = \$a;
```

```
$b = 666;
say "$a $b $b->$*";     # Error, ‹$b› is not a reference anymore
```

Dereferencia vráti *l-value*, do ktorej môžeme priradiť inú hodnotu:

```
$b->$* = 666;
say "$a $b $b->$*";    # 666 SCALAR(0x…) 666
```

Referencie: We need to go deeper

Referencie môžu mať viac než jednu úroveň:

```
my $x = 4;
my $ref2 = \\$x;

$ref2->$*->$* = 8;
```

Referencie: Operátor ‹ref›

```
ref EXPRESSION
```

Vráti typ referencie:

‹""› (nepravdivá hodnota) ‹EXPRESSION› nie je referencia

‹"SCALAR"› ‹EXPRESSION› je referencia na skalár

‹"CODE"› ‹EXPRESSION› je referencia na funkciu

‹"ARRAY"› ‹EXPRESSION› je referencia na pole

‹"HASH"› ‹EXPRESSION› je referencia na hash

‹"REF"› ‹EXPRESSION› je viacúrovňová referencia

(Ďalšie niekedy neskôr)

Referencie: Zoznam

Referencia na zoznam **neexistuje**.

Ak operátor ‹\› aplikujeme na zoznam, dostaneme zoznam referencií na prvky. Nasledujúce výrazy sú ekvivalentné:

```
\($x, $y, @z)
(\$x, \$y, \@z)
```

Referencie: Anonymné pole

```
my $array_ref = [ LIST… ];
```

Vytvorí nové pole zo zoznamu ‹LIST› a vráti naňho referenciu.

Referencie: Anonymné asociatívne pole

```
my $hash_ref = { LIST… };
```

Vytvorí nový hash zo zoznamu ‹LIST› a vráti naňho referenciu.

Blok alebo anonymný hash?

Perl sa podľa kontextu snaží uhádnuť, čo ‹{…}› znamená.
Môžeme to však (podľa potreby) vyjadriť explicitne:

- ‹+{ … }› Hash
- ‹{; … }› Blok

Hlboké štruktúry

Pole aj hash môžu ako hodnoty obsahovať len skaláry.
Tieto však môžu byť tiež referenciami:

```
my $books = [
    {
        title => 'The Fellowship of the Ring',
        authors => ['J. R. R. Tolkien'],
    },
];

say $books->[0]->{title};           # The Fellowship of the Ring
say $books->[0]->{authors}->[0];    # J. R. R. Tolkien
```

Referencie: Funkcie

```
sub foo($arg1, $arg2) {
    ...;
}

my $ref = \&foo;
```

Tieto referencie môžeme predávať ako parametre iným funkciám.

Circumfixová notácia

```
&$ref(ARGS…)
&{$ref}(ARGS…)
```

Postfixová notácia

```
$ref->(ARGS…);
```

Odbočka: Čo je ‹&›?

‹&› je *sigil* podobne ako ‹$›, ‹@› a ‹%›, akurát sa obvykle písať nemusí.

```
foo(1);
foo 1;
&foo(1);

# &foo 1;       # This does not work.
```

Má ‹&› niekedy zmysel okrem referencií?

Anonymné funkcie, uzávery

Anonymné funkcie

Ak použijeme ‹sub› bez názvu funkcie, vytvoríme anonymnú funkciu.

```
my $inc = sub ($a) { $a + 1 };
```

Výsledkom je *referencia* na funkciu.

Pripomenutie
Ak funkcia nemá ‹return›, vráti hodnotu posledného výrazu.
Používajte (ak vôbec) len u funkcií s jediným výrazom v tele.

Anonymné funkcie

Anonymné funkcie môžeme rovno použiť ako parameter inej funkcie:

```
sub apply($v, $f) {
    $f->($v)
}

apply(42, sub ($n) { $n + 1 });
```

Anonymné funkcie: Uzávery

Anonymné funkcie môžu pristupovať k premenným vo vonkajšom rozsahu:

```
sub power_f($exponent = 2) {
    return sub ($n) { $n ** $exponent };
}

say (power_f->(5));     # 25
say (power_f(3)->(5));  # 125
```

Špeciality

Špeciality: Anonymné rekurzívne funkcie

Lebo prečo nie.

Referenciu na aktuálnu funkciu Perl sprístupní v symbole `__SUB__`.

```
sub $fact = sub ($n) {
    return if $n < 0;
    return 1 if $n <= 1;
    return $n * __SUB__->($n);
};
```

Špeciality: Simulované viacrozmerné pole

Viacrozmerné pole
Táto vlastnosť je od ‹v5.36› vo východzom stave vypnutá

```
use feature qw{multidimensional};

my %canvas;
$canvas{$x, $y, $z} = $pixel;
```

Táto syntax je len skratka za

```
$canvas{ join($;, $x, $y, $z) } = $pixel;
```

Špeciality: Elipsa

Ak sa `...` nachádza v tele funkcie na mieste výrazu, tak pri vyhodnotení spôsobí chybu `Unimplemented`:

```
sub stub() {
    ...
}

stub;           # Dies with "Unimplemented".
```

Tento operátor sa tiež volá *yada yada*.

Operátor `A ... B` v zoznamovom kontexte funguje rovnako ako `A .. B`, v skalárnom kontexte sa volá *flip-flop* (uvidíme neskôr).