

petlja - bater

FUNKCIJE

(1) print "Ovo je neki tekst! \n" ili print 'Ovo je neki tekst! \n'

- dijagnostički učinak će interpretirati znak \n i prijeti u novi red,

dok jednostavni učinak neće interpretirati \n, nego ispraviti \n sa tekstrom

17/120. : 1007 oktalna ASCII vrijednost

1x7f heksadekadška ASCII vrijednost

print 5 * 4 : 20

print 5 * 4 : 5555 / ako je broj pokrivajući u cijeli broj, reducirat će se u cijeli : 3,99 je 3,9

print "5 * 4" : 5 * 4 / a ako je broj < 1, dobit ćeemo prazan rezultat

print "5 * 4" : 5 * 4

fa = 2; fb = 4; print fa * fb : 8

print "fa * fb" : 2 * 4 / provede, ali ne izracune !)

print 'fa * fb' : fa * fb

print "fa" * "fb" : 8

print "Rezultat je ". 6 * 7 . ".\n";

je isto što i print "Rezultat je ", 6 * 7, ".\n";

- ako stavimo točku (.)iza umnožka, mora biti razmak između ?

- operatori usporedbe

ASCII vrijednost : 0 - 48, A - 65, a - 97

=, !=, <, >, <=, >= za brojeve

(eg, ne, lt, gt, le, ge za značajne uzoče !)

" gt " : true

- logičke vrijednosti : nedefinisane vrijednost - false

null - false, ostali brojevi - true

prazan rezultat - false, ostali - true

(izuzuka je rezultat '0', - logička vrijednost je false !)

32/120.

(1)

```
$text = <STDIN>;
```

```
chomp($text);
```

```
print $text;
```

```
print "bla"
```

- učitamo tekst, chomp uključuje sa kraja ;

ispisuje učitani redak i bla u istom redak

- da neće chompa, nakon što bi program

ispisao učitani redak, otisao bi u novi

redak, i u novom redak ispisao bla

- nedefinirana vrijednost index :

- prije ugo se varijabli prvi put dodijeli vrijednost,

ona prima početnu vrijednost - index

(ako se radi o znotvorenoj listi - nizu - niz je,

ako je to u nizu broja - niz je vrijednost 0.)

listi i polja

- lista je skup podataka / niz je skup svih vrijednosti)

- polje je varijabla koja sadrži listu / koja može biti i prazna)

- indeksi počinju od 0

- ako se indeksira element iza kraja polja, njegova vrijednost je index!

@a = qw / fred barney betty wilma dino /;

print @a; : ispiše sve riječi, ali bez razmaka

print "@a" : ispiše polje riječi sa razmakinama

(2) push - dodaje element na kraj polja

pop - uključi zadnji element iz polja / ako je polje prazno,

shift - skida prvi element polja

vraca index, i polje ostaje prazno

unshift - dodaje element na početak polja

@fred = gw(hello dolly);

\$y = 2;

\$x = "This is \$fred[\$y-1]'s place";

print \$x : This is dolly's place

\$y = 2*4;

print \$x : This is 's place

\$y = "2*4"

print \$x : This is dolly's place

@fred = gw(eating rocks is wrong);

\$fred = "right";

print "this is \$fred[3]!a"; # wrong

print "this is \$fred[3]!a"; # right [3]

print "this is \$fred". "[3]!a"; # right [3]

print "this is \$fred[0]!a"; # right [3]

(3) - foreach \$variable (@polje) { ... }

(ako uema variable, utvare se podstavljuje vama variable \$-)

(4) - reverse

@obrnutia = reverse (@lista);

(ako se rezultat ne postavi u uku variable, gubi se !)

(5) sort

@sortirana = sort (@lista);

- ue vrijedi za brojce

- ue uječe u svoj argument (lista)

(ako se rezultat ne postavi u uku variable, gubi se !)

@ backwards = reverse gw / yabba, dabba doo);

print "@ backwards" : doo dabba yabba

\$ backwards = reverse \$w / yabba dabba doo);

print \$ backwards : oodabbaadabbay

@ backwards = reverse gw / yabba, dabba, doo);

print "@ backwards" : doo dabba, yabba,

@ backwards = reverse gw / yabba, dabba, doo);

print "@ backwards" : doo dabba yabba,

@ backwards = reverse gw / yabba, dabba, doo,);

print "@ backwards" : doo, dabba, , yabba,

print @ backwards : doo, dabba, yabba,

@ wilma = undef; # lista / undefined) { nije 1870 !

@ betty = (), # prazna lista

@ rocks = gw / talc quartz jade obsidian);

scalar @ rocks is @ rocks.

ispisuje broj elemenata liste (skalar),

a @ rocks ispisuje cijelu listu

65/110. ~~ju učinak funkcije je globalna varijable ; i to vrijedi
vrijek osim ako funkcija ne neglaziraju varijable sa my
tako da se postaje privatna! P)~~

- lista argumentata se poštjuje u @ (privatne varijable . P)

- prvi argument poštovan u \$-{0}, drugi u \$-{1}

(@max(10, 15), prvi funkcije s 2 argumenta
može se i postaviti (ako nije dozvoljene skraćenice))

Private variable

my (\$num) = @_; # \$num poprima vrijednost prveg elementa

my \$num = @_; # broj elemenata

74/120. - i; # povratak vrijednost / ako dođe do pogreške, vratiće -1)

- Perl poznaje bitu argumentata u posebnu varijablu @_ (potprogram?)
- argumenti navedeni u naredbenom rečniku pri pozivu programa,
u Perl programu dostupni su preko posabilog polja @ARGV ▷

80/120.

print sort <>;

- sortira ulazni niz učitani sa tipkovnice

- prelidanje učitavanje sa **Ctrl+D**, **<Enter>**, i on ispisuje sortirani niz

(6) print "%6d\n", 42; : 4 2

print "%2d\n", 434; : 4 3 4

print "%0-15s\n", "flintstone"; : flintstone 000000000000000

print "%10f\n", 6*7+2/3; : 42.666667

print "%10.3f\n", 6*7+2/3; : 42.667

print "%10.0f\n", 6*7+2/3; : 42.666667

print "%.1f\n", 6*7+2/3; : 42.666667 (9 znamenki!) ▷

Datoteke

open 86/120. - prijem / doći će do greške ako je datoteka postavljena
već sa rukom <> ili >>?)

- ako otvorimo datoteku za pisanje ili operisivanje, a ne postoji, stvoriće se
tako da će otvaranje ujek biti uspešno

- obim u slučaju otvaranja uspostavljaće datoteku za čitanje

- tada je već otvoren, i otvaranje već je bilo uspešno

Asoociativno polje - hash

- indeksiraju elementa pomoći fiksiranog uiza zatvara - ključ
(proizvodnji skup, koji se pretvara u string?)

- veličine asoc. polja nije ogranicena ▷ ispis asoc. polja

while(\$key, \$value) = each %hash {
 print "\$key => \$value\n";

- pristup elementu abac polja : " → " unijeto zapisu

\$hash{ \$some - key };

- pristup abac polju s nepostojecim indeksom vraca vrijednost undef!

- vrijednost abac polja u kontekstu liste

(lista parova ključ - vrijednost.)

@ my - array = % some - hash;

- ako vrijednosti nisu jedinstvene, duplicitni elementi se prepoznaju

(ostaje zadje zapisana vrijednost) prilikom operacije reverse!

- ali i za ispisom potroci (each) parova ključ - vrijednost

se također pabe duplicitni elementi, ostaje zadje zapisan

@ k = keys % hash; # vraca listu ključeva abac polja

- @ v = values % hash; # lista vrijednosti

(%key, %value) = each % hash; # par ključ - vrijednost

- exists : preverava postojanje nekog ključa u polju

(vraca boolovu vrijednost - 0 false, ili 1 true)

101/120. Unosi učinkovitosti iz polja - delete \$hash{"key"};

105/120. (1a10) - proizvoljni znak uključujući "/u" / za razliku od ".")

Znamente 0-9 + sve one što nisu znamente

sidla / anchors)

1b - granica riječi definirane kao prazni niz između znaka riječi

/1w : A-2a-20-9-) ; znaka koji nije znak riječi /1w)

- proizvoljnim redoslijedom.

106/120.

- ?

\$ = "/u";

if (/s) {

print "H' matched '/u";

podudara se, ali

if (/I/) { ... } ue ?

1B : učinkovitost od 1b

109/120. opije podudaraju 1s

{ ^ pocetak retka
 \\$ kraj retka

! 110/120. 1g my \$text = "per pe blaper perl";
my @words = (\$text =~ /\per/gi);
print "Result : ", scalar @words, "\n";

112/120. Naslitesacija

\$cut = tr /~/~1/;

\$cut = tr /0-9//;

! - " ~ fled a ** ** ";

\$cut = tr /~/~/~1/;

print \$cut; #5

!

\$bla = "AaAAbCC"

\$bla = ~ tr /A-Z/a-z/;

print "\$bla"; # aaabcc

tr/lista-pretrazivajja / lista-zamjene
- ako je uiz zamjenstih znakova
prazan, nema zamjene?

variable podudaraju

- variable koja pamt proizvorne poduzeve \$1, \$2, \$3, ...

Povezivaju ulaz - saček reference 10, 12, 13, ...

- predmetni poduzi može se koristiti za podudaraju
u ostalim izlaza

11(w|w|w)1s1/; # naziv polovjene imjave riječi u tekstu

Uzorak za podudaraju

10 : A-Za-z0-9.

1s : razni znak (/f, lt, lu, lr)

116/120.

(7)

split - dijeli ulazni uiz ne separatorima

podrazumijevani oblik default) je dijeljenje ne
praznimama - 1s+ !

join - povezuje ulaznu listu u jedan zatvoreni uiz
unutarjuenog znakice

- da bi se broj znakova okatalijun, mora :

a) zapovijati 5 10

b) znati 1 do 3 znakove (ja vrijednost mora biti manja od 256 (jedan oktet))

c) ako broj imat 3 znakove, prve dve se mijenjaju mreže
priut "10000007";

če ispisati $\ll 0007$;

- utima samo prve dve mreže uaku 10, a 007 se nekita potrebno

- ispisivanjem mrež karaktera dobijemo prazan znak (razmak)

logičke vrijednosti:

- u feli je ugradeno da se znakovni uiz '0' tretira kao false

priut '0' + 2; #2

jer će se uiz koji uje broj pretvoriti u vrijednost 0

(mlikom unutarnje operacija?)

@ rocks = gw / tale quartz jade obsidian;

priut "I have ", @ rocks, $\ll \ll$ rocks.1u";

#ovo je "neodlučan" kontekst, pa se lista pretvara u string (ispis bez razmaka)

priut "I have ", "rocks", "rocks.1u";

#duostuki uvedući interpolacija - pa se izlucuju elementata liste

ubacuje sadržaj variable \$", što je po defaultu razmak

priut "I have ", scalar @ rocks, "rocks.1u";

#scalar formira skalaru vrijednost, te se zbroj tog i ispisuje br. elementata

priut "I have ", @ rocks(), "rocks.1u";

zatijeva skale, točnije string, i s lijeve, i s

desne strane. Budući da je s lijeve strane lista,

mo se ona pretvara u skalar (a to je po definiciji moj

elementata u listi), pa se pretvara u string i dodaje na

uiz sa desne strane, a to je! rocks.