Ukazni programski jezik

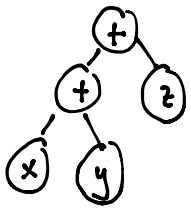
Aritmetični izrazi

konkretna sintaksa

```
⟨aritmetični-izraz⟩ ::= ⟨aditivni-izraz⟩

⟨aditivni-izraz⟩ ::= ⟨multiplikativni-izraz⟩ | ⟨aditivni-izraz⟩ + ⟨multiplikativni-izraz⟩

⟨multiplikativni-izraz⟩ ::= ⟨osnovni-izraz⟩ | ⟨multiplikativni-izraz⟩ * ⟨osnovni-izraz⟩

⟨osnovni-izraz⟩ ::= ⟨spremenljivka⟩ | ⟨številka⟩ | ( ⟨aritmetični-izraz⟩ )

⟨spremenljivka⟩ ::= [a-zA-z]+

⟨številka⟩ ::= -? [0-9]+
```

"x+y+z"

"(x+y)+z"



Boolovi izrati:

abstraktna sintaksa

```
⟨boolov-izraz⟩ ::= true | false |
                   ⟨aritmetični-izraz⟩ = ⟨aritmetični-izraz⟩ |
                   ⟨aritmetični-izraz⟩ < ⟨aritmetični-izraz⟩ |
                   ⟨boolov-izraz⟩ and ⟨boolov-izraz⟩ |
                   ⟨boolov-izraz⟩ or ⟨boolov-izraz⟩ |
                   not ⟨boolov-izraz⟩
```

3<5 and 7=7 or 5<10

boolov izr. or boolov. izr.

3<5 and 7=7 or 5<10

boolov and boolov izr.

Podamo še prioriteto in asociativnost operacij:

- or (levo) ⇒ 3<5 and 7<7 or 5<10
- and (levo)

"and ima prednost pred or"

P and Q and R levo asociativen

Dodati bi morali še oklepaje: (⟨boolov-izraz⟩)

Ukazi:

```
⟨ukaz⟩ ::= skip |
           ⟨spremenljivka⟩ := ⟨aritmetični-izraz⟩ |
           ⟨ukaz⟩ ; ⟨ukaz⟩ |
           while ⟨boolov-izraz⟩ do ⟨ukaz⟩ done |
           if ⟨boolov-izraz⟩ then ⟨ukaz⟩ else ⟨ukaz⟩ end
```

skip — ne naredi ničesar (Python: pass)

$x := e$ — nastavi x na e

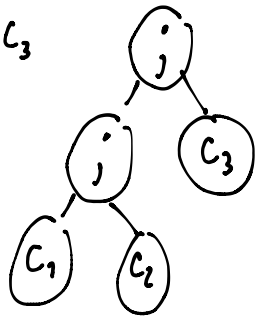
$c_1 ; c_2$ — naredi c_1 in potem c_2

while b do
  c_1;
  c_2;
  c_3
done

Prioritete:

* `;` (levo)
* `or` (levo)
* `and` (levo)
* `not`
* `<` `=`
* `+` (levo)
* `*` (levo)

$c_1 ; c_2 ; c_3 = (c_1 ; c_2) ; c_3$



$x + 7 < 5$

$x + (7 < 5)$ NAROBE

ni aritmetični

Operacijska semantika ukazov

Aritmetični izrazi: $\eta \mid e \hookrightarrow n$

V okolju η se izraz e evaluira v število n

izračuna

$$\eta \mid e \longmapsto e'$$

V okolju η izraz e naredi en računski korak in postane e'

Java:

`++n + 1`

aritmetični izraz, <u>spremeni</u> okolje (polica n)

<u>Ukazi</u>:

$$(\eta, C) \longmapsto (\eta', C')$$

V okolju η ukaz C naredi en korak izvajanja in dobimo (morda spremenjeno) okolje η' ter ukaz C' (nadaljevanje izvajanja)

$$(\eta, C) \longmapsto \eta'$$

V okolju η se ukaz C konča v enem koraku in dobimo okolje η'.

<u>Zapis</u>:

$\eta[x \mapsto n]$ okolje, ki deluje kot η, le da x slika v n

Primer: $\eta = [x \mapsto 7, y \mapsto 10, a \mapsto 0]$

$\eta[y \mapsto 5] = [x \mapsto 7, y \mapsto 5, a \mapsto 0]$

(glej operacijsko semantiko v zapiskih ...)

Primer:

$([x \mapsto 1, y \mapsto 2], (\text{if } x < y+3 \text{ then } x := y+2 \text{ else skip end})) \mapsto$

$[x \mapsto 1, y \mapsto 2] \mid x < y+3 \hookrightarrow \text{true}$

$([x \mapsto 1, y \mapsto 2], x := y+2) \mapsto$

$[x \mapsto 1, y \mapsto 2] \mid y+2 \hookrightarrow 4$

$[x \mapsto 4, y \mapsto 2]$

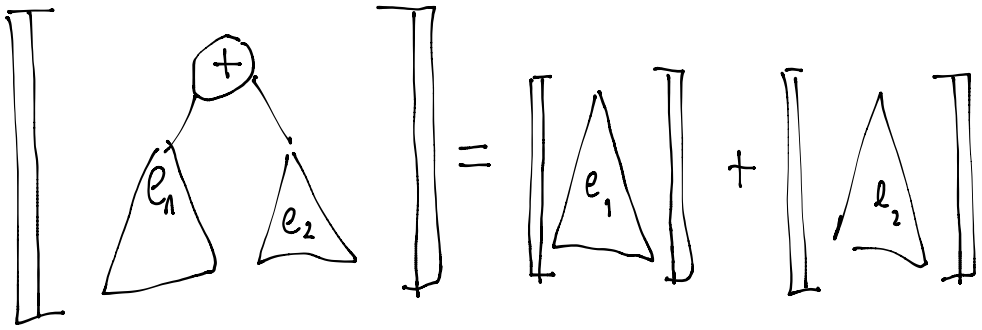
Denotacijska semantika

"42" 42

$[\![42]\!] = 42$

↑ izraz ↑ "pomeni" ↑ matematični objekt

!!



Ekvivalenca programov

(glej zapiske)

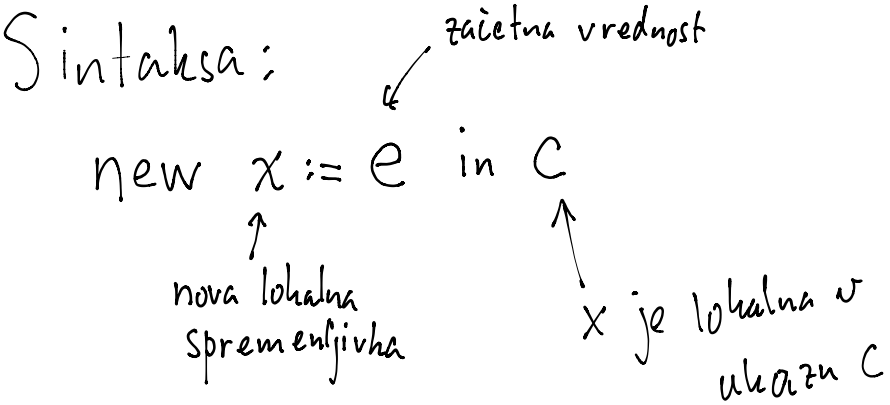
Nova lokalna spremenljivka

```
if ( ... ) {
        int x = 7;
} else {
    ⋮
}
```

← lokalna spremenljivka

vidna v svojem bloku

Sintaksa:



Primer:

```
new i := 1 in
  (new s := 0 in
     while i < 101 do
       s := s + i ;
       i := i + 1
     done)
```

```
new i := 1 in
new s := 0 in
  while i < 101 do
    s := s + i ;
    i := i + 1
  done
```