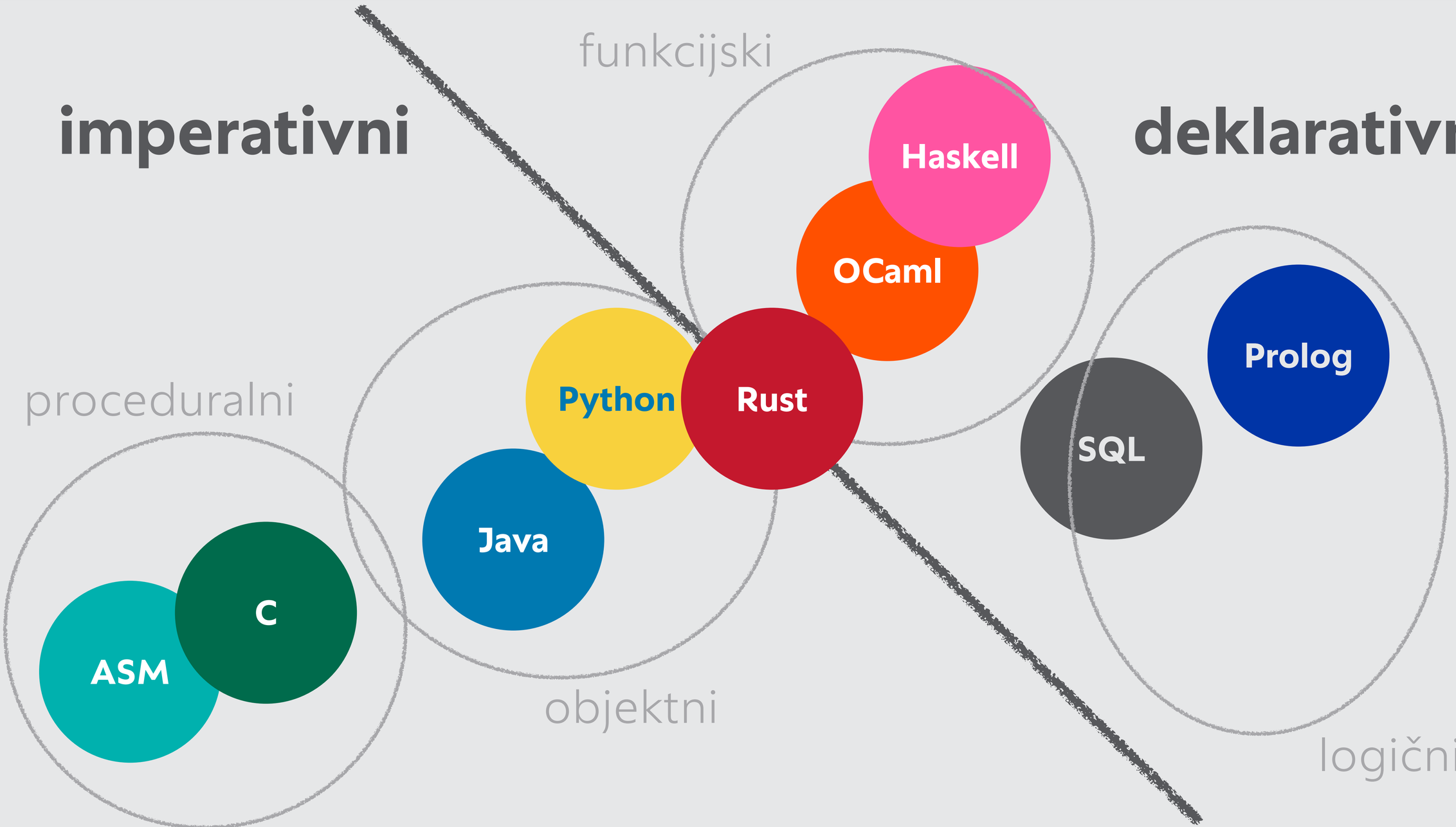
Ukazni programski jezik

prejšnjič...

Programski jeziki se med seboj lahko zelo **razlikujejo**

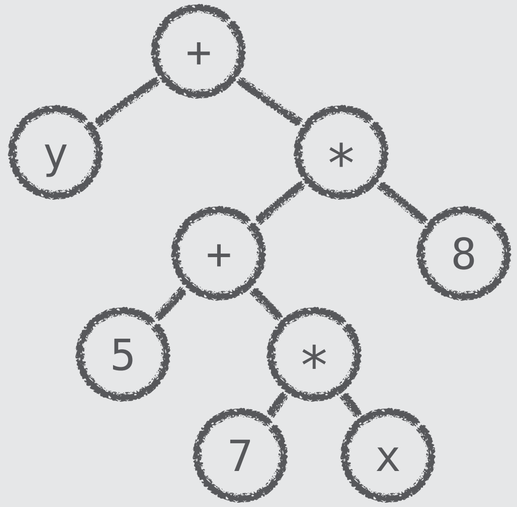


Ločimo med **konkretno** in **abstraktno** sintakso

"y + (5+7 * x) * 8"

```
SPREMENLJIVKA(y) PLUS OKLEPAJ
 ŠTEVILKA(5) PLUS ŠTEVILKA(7)
KRAT SPREMENLJIVKA(x) ZAKLEPAJ
     KRAT ŠTEVILKA(8) EOF
```

```
          ⟨izraz⟩ ::= ⟨aditivni⟩ EOF
       ⟨aditivni⟩ ::= ⟨multiplikativni⟩
                    | ⟨aditivni⟩ PLUS ⟨multiplikativni⟩
⟨multiplikativni⟩ ::= ⟨osnovni⟩
                    | ⟨multiplikativni⟩ KRAT ⟨osnovni⟩
        ⟨osnovni⟩ ::= ⟨spremenljivka⟩
                    | ⟨število⟩
                    | OKLEPAJ ⟨aditivni⟩ ZAKLEPAJ
  ⟨spremenljivka⟩ ::= SPREMENLJIVKA(string)
       ⟨število⟩ ::= ŠTEVILKA(int)
```



$$e ::= n \mid x \mid e_1 + e_2 \mid e_1 * e_2$$

Semantiko jezika podamo z relacijo **velikih korakov**

$\eta \mid e \hookrightarrow n$

$$\frac{\eta(x) = n}{\eta \mid x \hookrightarrow n} \qquad \frac{}{\eta \mid n \hookrightarrow n}$$

$$\frac{\eta \mid e_1 \hookrightarrow n_1 \qquad \eta \mid e_2 \hookrightarrow n_2 \qquad n_1 + n_2 = n}{\eta \mid e_1 + e_2 \hookrightarrow n}$$

$$\frac{\eta \mid e_1 \hookrightarrow n_1 \qquad \eta \mid e_2 \hookrightarrow n_2 \qquad n_1 \cdot n_2 = n}{\eta \mid e_1 * e_2 \hookrightarrow n}$$

Izberemo lahko tudi semantiko **malih korakov**

$\eta \mid e_1 \mapsto e_2$

$$\frac{\eta(x) = n}{\eta \mid x \mapsto n}$$

$$\frac{\eta \mid e_1 \mapsto e_1'}{\eta \mid e_1 + e_2 \mapsto e_1' + e_2} \qquad \frac{\eta \mid e_2 \mapsto e_2'}{\eta \mid n_1 + e_2 \mapsto n_1 + e_2'} \qquad \frac{n_1 + n_2 = n}{\eta \mid n_1 + n_2 \mapsto n}$$

$$\frac{\eta \mid e_1 \mapsto e_1'}{\eta \mid e_1 * e_2 \mapsto e_1' * e_2} \qquad \frac{\eta \mid e_2 \mapsto e_2'}{\eta \mid n_1 * e_2 \mapsto n_1 * e_2'} \qquad \frac{n_1 \cdot n_2 = n}{\eta \mid n_1 * n_2 \mapsto n}$$

Sintaksa in semantika

Sintakso jezika ločimo na **aritmetične** in **logične** izraze ter **ukaze**

aritmetični izraz $e ::= x \mid n \mid e_1 + e_2 \mid e_1 * e_2$

logični izraz $b ::= \texttt{true} \mid \texttt{false} \mid e_1 = e_2 \mid e_1 < e_2 \mid$
$b_1 \texttt{ and } b_2 \mid b_1 \texttt{ or } b_2 \mid \texttt{not } b$

ukaz $c ::= \texttt{skip} \mid x := e \mid c_1; c_2 \mid$
$\texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2 \texttt{ end} \mid$
$\texttt{while } b \texttt{ do } c \texttt{ done}$

Program, ki **sešteje prvih 100** naravnih števil

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

Semantiko **aritmetičnih izrazov** podamo z **velikimi koraki**

$\eta \mid e \hookrightarrow n$

$$\frac{\eta(x) = n}{\eta \mid x \hookrightarrow n} \qquad \frac{}{\eta \mid n \hookrightarrow n}$$

$$\frac{\eta \mid e_1 \hookrightarrow n_1 \qquad \eta \mid e_2 \hookrightarrow n_2 \qquad n_1 + n_2 = n}{\eta \mid e_1 + e_2 \hookrightarrow n}$$

$$\frac{\eta \mid e_1 \hookrightarrow n_1 \qquad \eta \mid e_2 \hookrightarrow n_2 \qquad n_1 \cdot n_2 = n}{\eta \mid e_1 * e_2 \hookrightarrow n}$$

Semantiko **logičnih izrazov** podamo podobno

$\eta \mid b \hookrightarrow v$

logični izraz $b ::= \texttt{true} \mid \texttt{false} \mid e_1 = e_2 \mid e_1 < e_2 \mid b_1 \texttt{ and } b_2 \mid b_1 \texttt{ or } b_2 \mid \texttt{not } b$

$$\frac{}{\eta \mid \texttt{true} \hookrightarrow \texttt{true}} \qquad \frac{}{\eta \mid \texttt{false} \hookrightarrow \texttt{false}}$$

$$\frac{\eta \mid e_1 \hookrightarrow n_1 \quad \eta \mid e_2 \hookrightarrow n_2 \quad n_1 = n_2}{\eta \mid e_1 = e_2 \hookrightarrow \texttt{true}} \qquad \frac{\eta \mid e_1 \hookrightarrow n_1 \quad \eta \mid e_2 \hookrightarrow n_2 \quad n_1 \neq n_2}{\eta \mid e_1 = e_2 \hookrightarrow \texttt{false}}$$

$$\frac{\eta \mid e_1 \hookrightarrow n_1 \quad \eta \mid e_2 \hookrightarrow n_2 \quad n_1 < n_2}{\eta \mid e_1 < e_2 \hookrightarrow \texttt{true}} \qquad \frac{\eta \mid e_1 \hookrightarrow n_1 \quad \eta \mid e_2 \hookrightarrow n_2 \quad n_1 \geq n_2}{\eta \mid e_1 < e_2 \hookrightarrow \texttt{false}}$$

Semantiko **logičnih veznikov** podamo kratkostično

$\eta \mid b \hookrightarrow v$

logični izraz $b ::= \texttt{true} \mid \texttt{false} \mid e_1 = e_2 \mid e_1 < e_2 \mid b_1 \texttt{ and } b_2 \mid b_1 \texttt{ or } b_2 \mid \texttt{not } b$

$$\frac{\eta \mid b_1 \hookrightarrow \texttt{true} \qquad \eta \mid b_2 \hookrightarrow v_2}{\eta \mid b_1 \texttt{ and } b_2 \hookrightarrow v_2}$$

$$\frac{\eta \mid b_1 \hookrightarrow \texttt{false}}{\eta \mid b_1 \texttt{ and } b_2 \hookrightarrow \texttt{false}}$$

$$\frac{\eta \mid b_1 \hookrightarrow \texttt{true}}{\eta \mid b_1 \texttt{ or } b_2 \hookrightarrow \texttt{true}}$$

$$\frac{\eta \mid b_1 \hookrightarrow \texttt{false} \qquad \eta \mid b_2 \hookrightarrow v_2}{\eta \mid b_1 \texttt{ or } b_2 \hookrightarrow v_2}$$

$$\frac{\eta \mid b \hookrightarrow \texttt{true}}{\eta \mid \texttt{not } b \hookrightarrow \texttt{false}}$$

$$\frac{\eta \mid b \hookrightarrow \texttt{false}}{\eta \mid \texttt{not } b \hookrightarrow \texttt{true}}$$

Semantiko **ukazov** bomo podali z **malimi koraki**

$$(\eta, c) \mapsto (\eta', c')$$

Semantiko **logičnih izrazov** podamo podobno

$(\eta, c) \mapsto (\eta', c')$

ukaz $c ::= \texttt{skip} \mid x := e \mid c_1; c_2 \mid \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \texttt{end} \mid \texttt{while}\ b\ \texttt{do}\ c\ \texttt{done}$

$$\frac{\eta \mid x \hookrightarrow n}{(\eta, x := e) \mapsto (\eta[x \mapsto n], \texttt{skip})}$$

kot η, samo x je nastavljen na n

$$\frac{(\eta, c_1) \mapsto (\eta', c_1')}{(\eta, c_1; c_2) \mapsto (\eta', c_1'; c_2)} \qquad \frac{}{(\eta, \texttt{skip}; c_2) \mapsto (\eta, c_2)}$$

$$\frac{\eta \mid b \hookrightarrow \texttt{true}}{(\eta, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \texttt{end}) \mapsto (\eta, c_1)} \qquad \frac{\eta \mid b \hookrightarrow \texttt{false}}{(\eta, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \texttt{end}) \mapsto (\eta, c_2)}$$

$$\frac{\eta \mid b \hookrightarrow \texttt{true}}{(\eta, \texttt{while}\ b\ \texttt{do}\ c\ \texttt{done}) \mapsto (\eta, c; \texttt{while}\ b\ \texttt{do}\ c\ \texttt{done})} \qquad \frac{\eta \mid b \hookrightarrow \texttt{false}}{(\eta, \texttt{while}\ b\ \texttt{do}\ c\ \texttt{done}) \mapsto (\eta, \texttt{skip})}$$

Jezik je določen s **sintakso** in **semantiko**

sintaksa
pravila pisanja

statična semantika
pravila veljavnosti

denotacijska semantika
matematični pomen

dinamična semantika
pravila izvajanja

ukazni jezik

COMM

Ekvivalenca programov

Kontekst izrazov je ukaz z eno **luknjo**, v katero spada izraz

$$\mathscr{C} = \texttt{s := 0;}$$
$$\texttt{i := 0;}$$
$$\texttt{while [] do}$$
$$\texttt{s := s + i;}$$
$$\texttt{i := i + 1}$$
$$\texttt{done}$$

Če luknjo **napolnimo** z ustreznim izrazom, dobimo ukaz

$$\mathscr{C}[\texttt{i < 101}] = \texttt{s := 0;}$$

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

Če luknjo **napolnimo** z ustreznim izrazom, dobimo ukaz

$$\mathscr{C}[\texttt{true}] = \texttt{s := 0;}$$

```
i := 0;
while true do
  s := s + i;
  i := i + 1
done
```

Kontekst ukazov je ukaz z luknjo, v katero vstavimo **ukaz**

$$\mathscr{C} = \texttt{s := 0;}$$
$$\texttt{i := 0;}$$
$$\texttt{while i < 101 do}$$
$$\texttt{s := s + i;}$$
$$\texttt{[]}$$
$$\texttt{done}$$

Če luknjo **napolnimo** z ustreznim ukazom, dobimo ukaz

$$\mathscr{C}[\![\ \texttt{i:=i+1}\ ]\!] = \texttt{s := 0;}$$

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

Če luknjo **napolnimo** z ustreznim ukazom, dobimo ukaz

$$\mathscr{C}[\![\ \texttt{skip}\ ]\!] = \texttt{s := 0;}$$

```
i := 0;
while i < 101 do
  s := s + i;
  skip
done
```

Kdaj dva izraza/ukaza obravnavamo kot **ekvivalentna**?

$$A \simeq B$$

v poljubnem kontekstu $\mathscr{C}$ in okolju η

za poljubno končno okolje η'

v poljubnem številu korakov

$$\forall \mathscr{C} . \forall \eta . \forall \eta' .$$

$$(\eta, \mathscr{C}[A]) \mapsto^{*} (\eta', \texttt{skip})$$

$\mathscr{C}[A]$ konča v η'

$$\iff$$

$$(\eta, \mathscr{C}[B]) \mapsto^{*} (\eta', \texttt{skip})$$

$\mathscr{C}[B]$ konča v η'

Sta spodnja izraza ekvivalentna?

1 $\not\simeq$ 2

$\eta = []$ $\mathscr{C} = (x := [])$

Sta spodnja ukaza ekvivalentna?

$$x := x + 1; \quad x := x + 2 \quad \simeq \quad x := x + 3$$

Sta spodnja ukaza ekvivalentna?

x := x + 1;
x := x + 2

$\not\simeq$

y := y + 3

$\eta = []$

$\mathscr{C} = []$

Sta spodnja ukaza ekvivalentna?

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

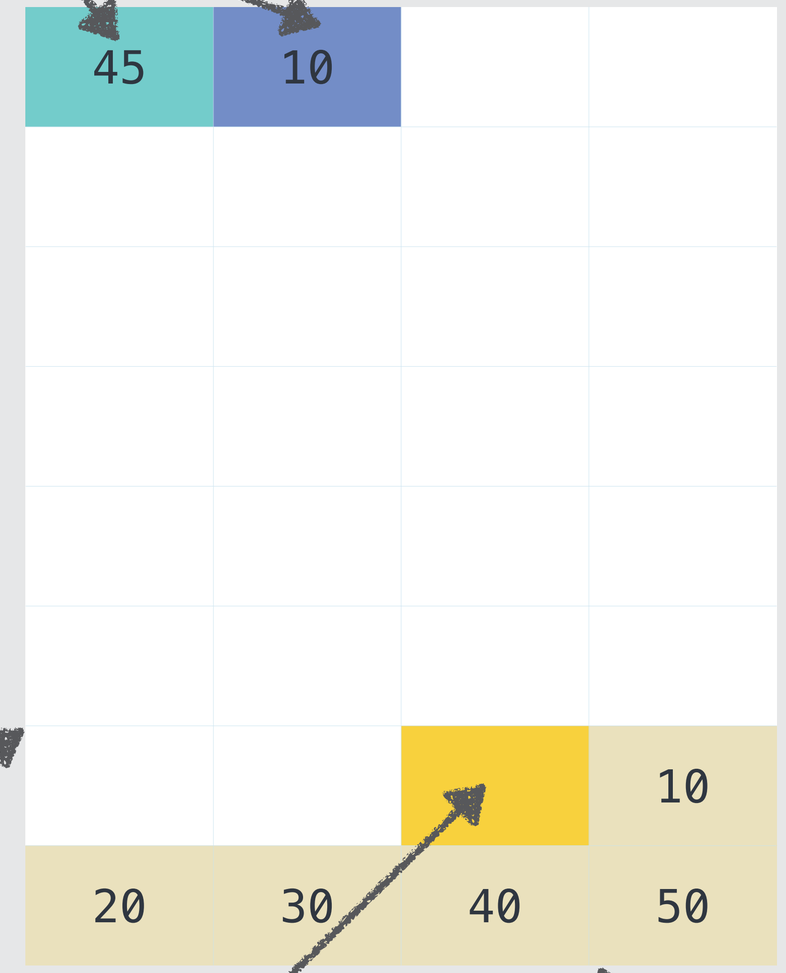
$\not\simeq$ `s := 5050`

manjka `i := 0`

Prevajalnik v strojno kodo

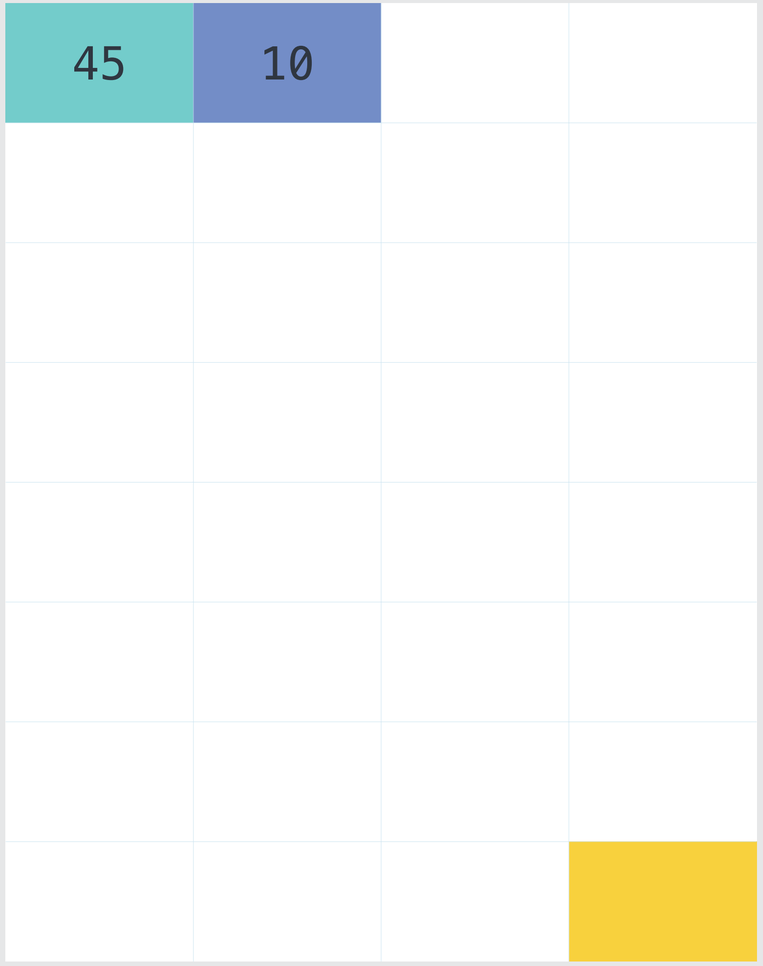
Ukazni jezik bomo izvajali na **virtualnem stroju**

```
000 PUSH 0
001 SET 0
002 PUSH 0
003 SET 1
004 GET 1
005 PUSH 101
006 LT
007 JMPZ 9
008 GET 0
009 GET 1
010 ADD
011 SET 0
012 GET 1
013 PUSH 1
014 ADD
015 SET 1
016 JMP -13
```



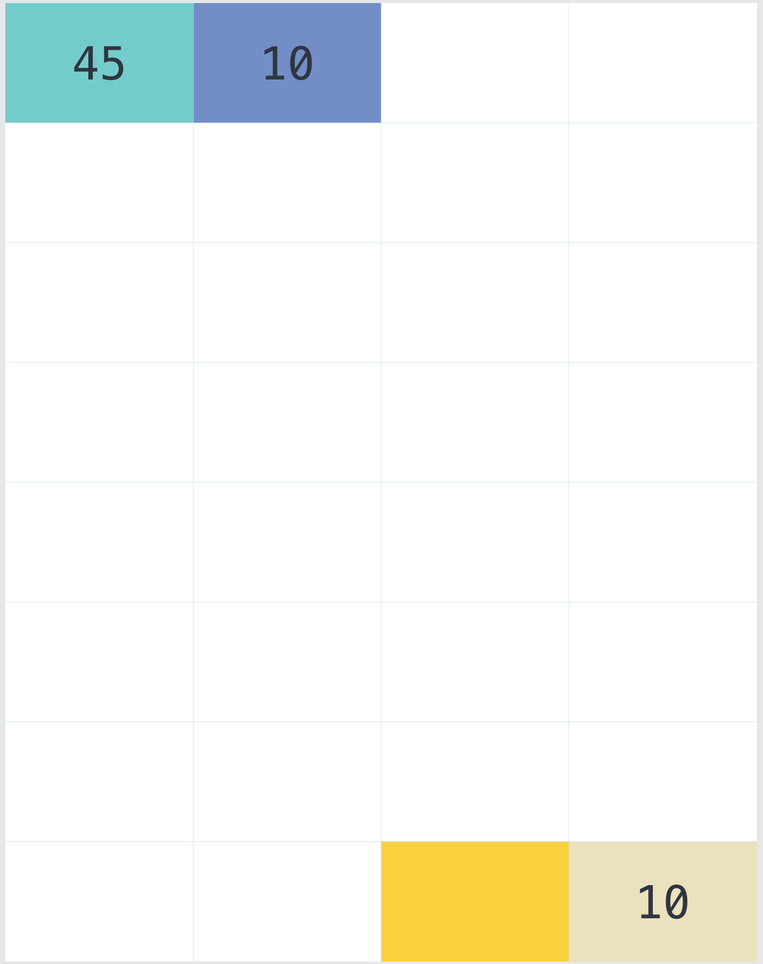
GET na vrh sklada **naloži** vsebino pomnilniške lokacije

```
000 PUSH 0
001 SET 0
002 PUSH 0
003 SET 1
004 GET 1
005 PUSH 101
006 LT
007 JMPZ 9
008 GET 0
009 GET 1
010 ADD
011 SET 0
012 GET 1
013 PUSH 1
014 ADD
015 SET 1
016 JMP -13
```



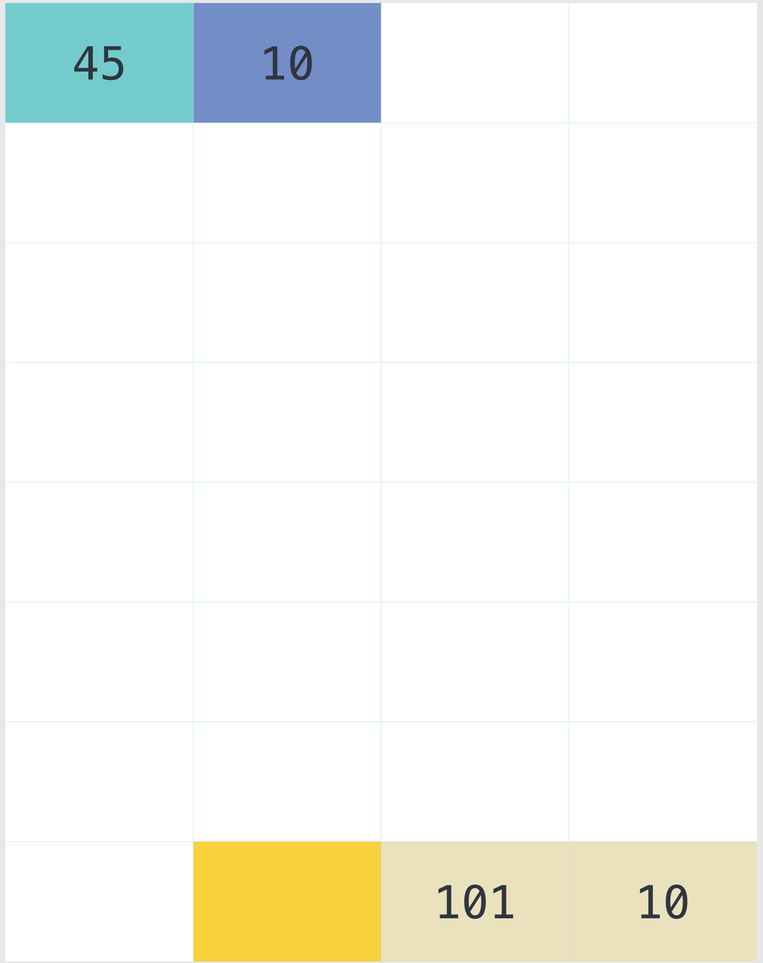
PUSH na vrh sklada doda konstanto

```
000 PUSH 0
001 SET 0
002 PUSH 0
003 SET 1
004 GET 1
005 PUSH 101
006 LT
007 JMPZ 9
008 GET 0
009 GET 1
010 ADD
011 SET 0
012 GET 1
013 PUSH 1
014 ADD
015 SET 1
016 JMP -13
```



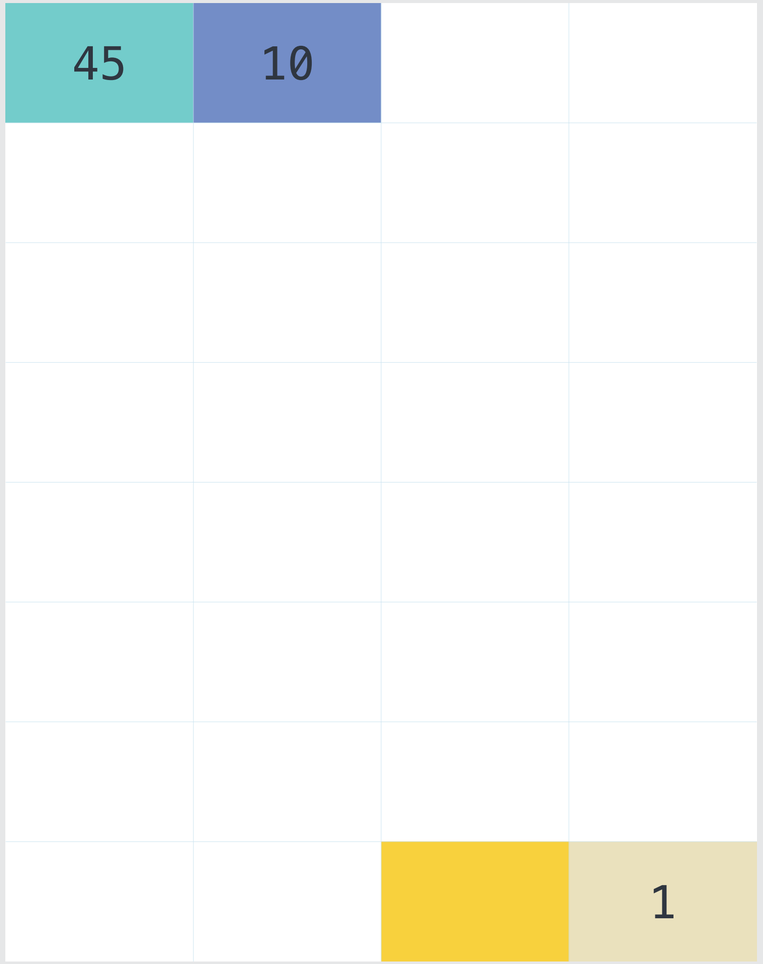
LT vrhnji dve vrednosti nadomesti z **rezultatom primerjave**

```
000 PUSH 0
001 SET 0
002 PUSH 0
003 SET 1
004 GET 1
005 PUSH 101
006 LT
007 JMPZ 9
008 GET 0
009 GET 1
010 ADD
011 SET 0
012 GET 1
013 PUSH 1
014 ADD
015 SET 1
016 JMP -13
```



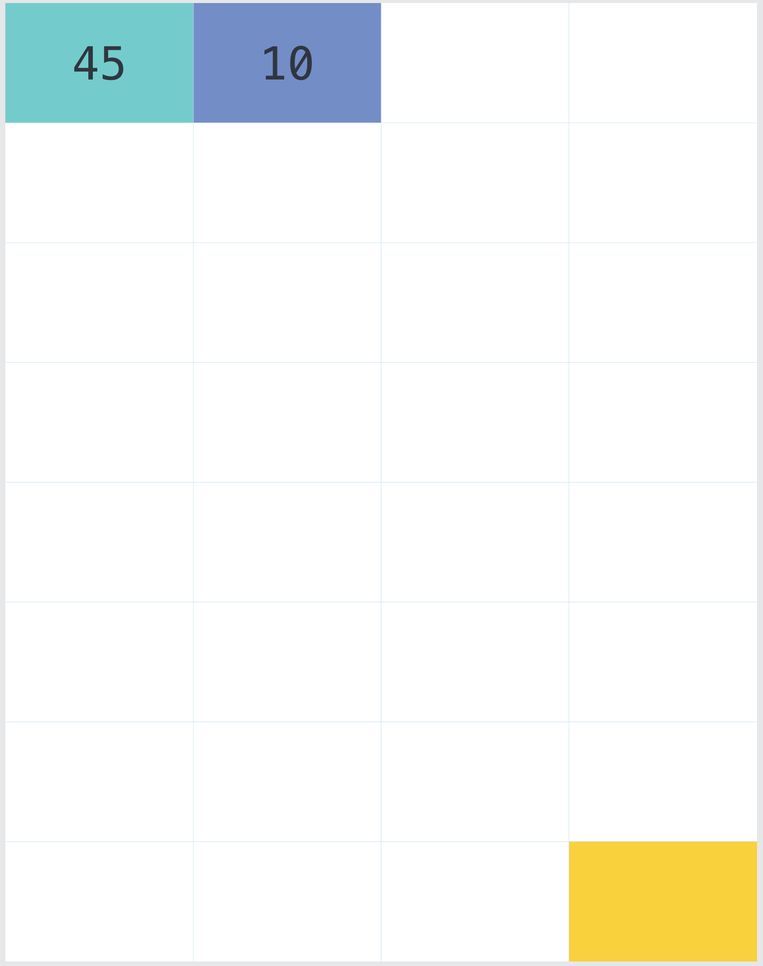
JMPZ naredi **relativni skok**, če je na vrhu ničla

```
000 PUSH 0
001 SET 0
002 PUSH 0
003 SET 1
004 GET 1
005 PUSH 101
006 LT
007 JMPZ 9
008 GET 0
009 GET 1
010 ADD
011 SET 0
012 GET 1
013 PUSH 1
014 ADD
015 SET 1
016 JMP -13
```



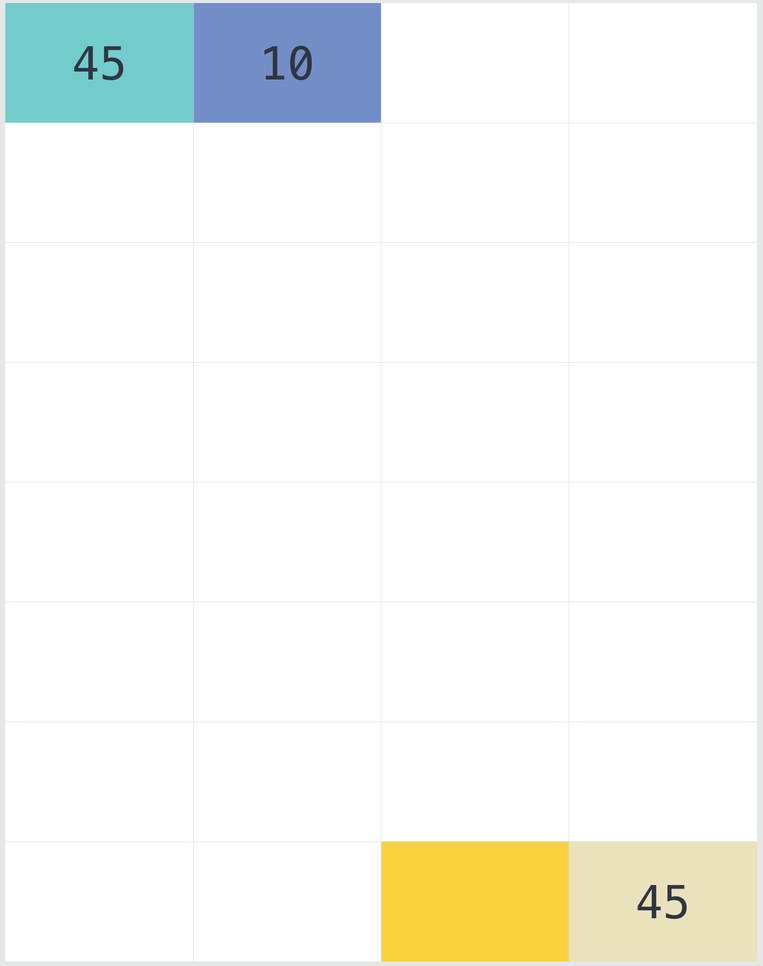
GET na vrh sklada **naloži** vsebino pomnilniške lokacije

```
000 PUSH 0
001 SET 0
002 PUSH 0
003 SET 1
004 GET 1
005 PUSH 101
006 LT
007 JMPZ 9
008 GET 0
009 GET 1
010 ADD
011 SET 0
012 GET 1
013 PUSH 1
014 ADD
015 SET 1
016 JMP -13
```



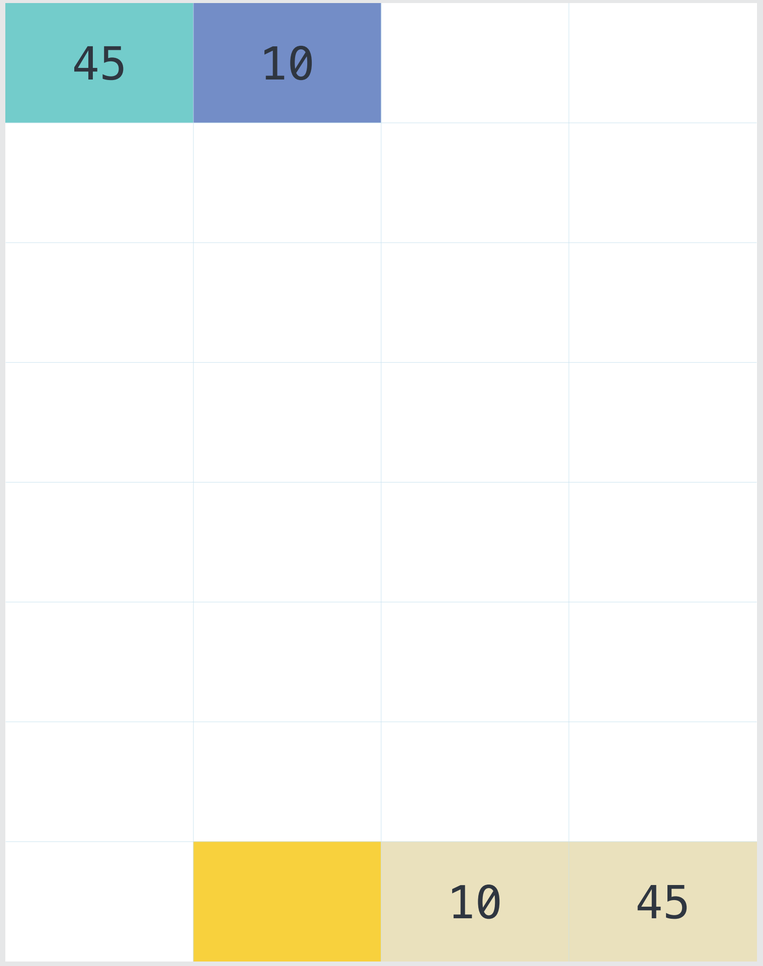
GET na vrh sklada **naloži** vsebino pomnilniške lokacije

```
000 PUSH 0
001 SET 0
002 PUSH 0
003 SET 1
004 GET 1
005 PUSH 101
006 LT
007 JMPZ 9
008 GET 0
009 GET 1
010 ADD
011 SET 0
012 GET 1
013 PUSH 1
014 ADD
015 SET 1
016 JMP -13
```



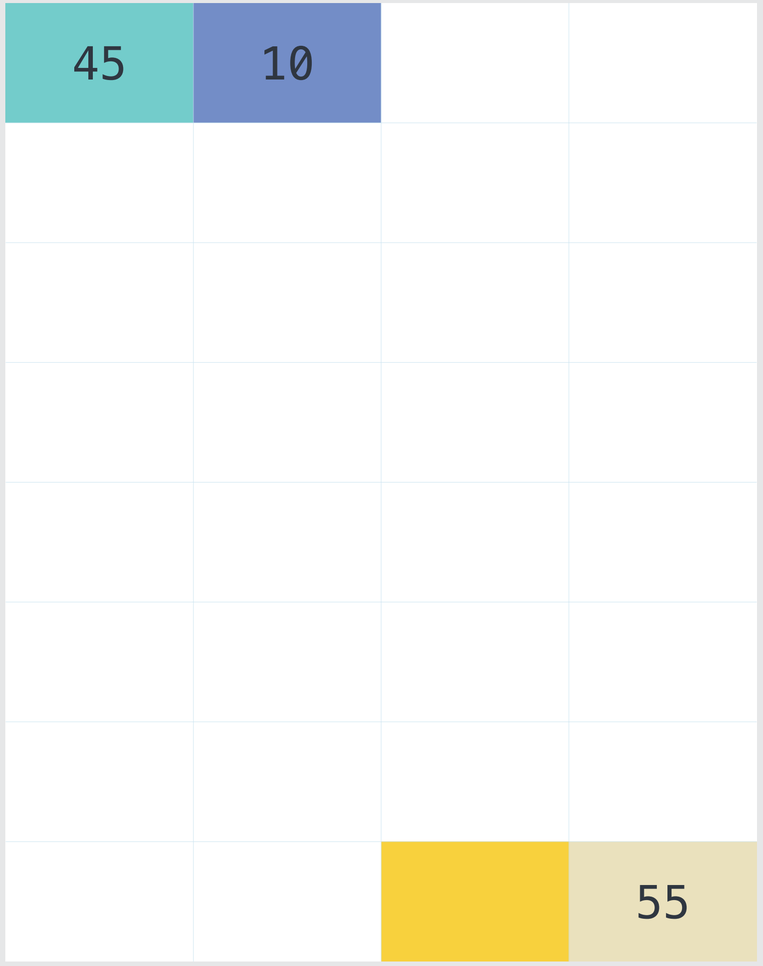
ADD vrhnji dve vrednosti nadomesti z **njuno vsoto**

```
000 PUSH 0
001 SET 0
002 PUSH 0
003 SET 1
004 GET 1
005 PUSH 101
006 LT
007 JMPZ 9
008 GET 0
009 GET 1
010 ADD
011 SET 0
012 GET 1
013 PUSH 1
014 ADD
015 SET 1
016 JMP -13
```



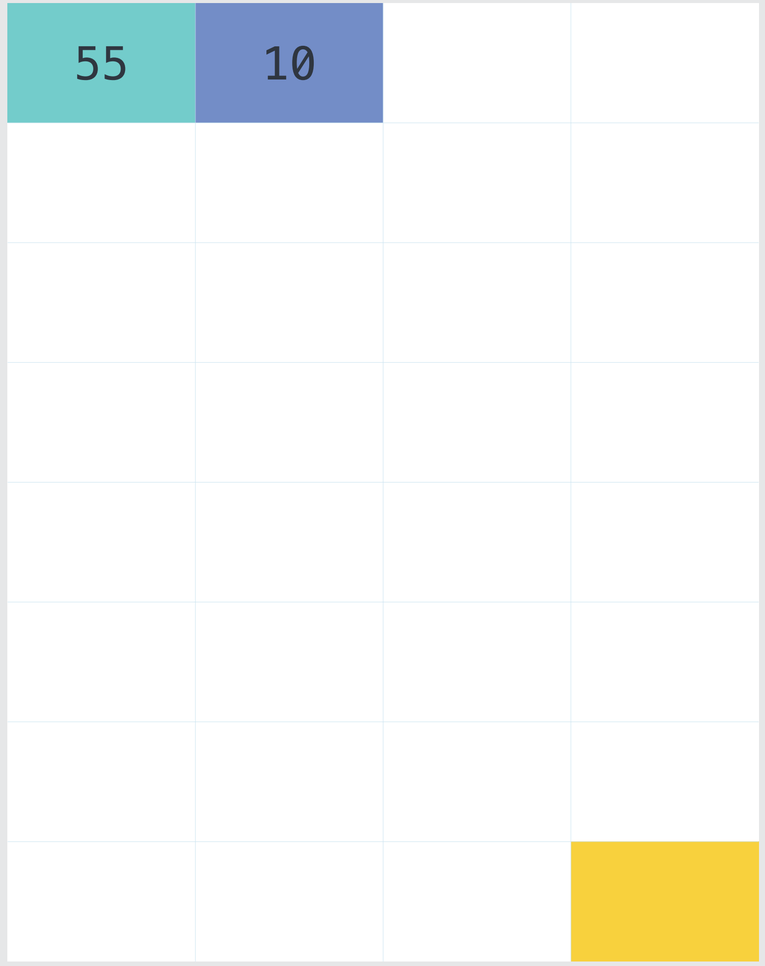
SET vrh sklada **shrani** v dano pomnilniško lokacijo

```
000 PUSH 0
001 SET 0
002 PUSH 0
003 SET 1
004 GET 1
005 PUSH 101
006 LT
007 JMPZ 9
008 GET 0
009 GET 1
010 ADD
011 SET 0
012 GET 1
013 PUSH 1
014 ADD
015 SET 1
016 JMP -13
```



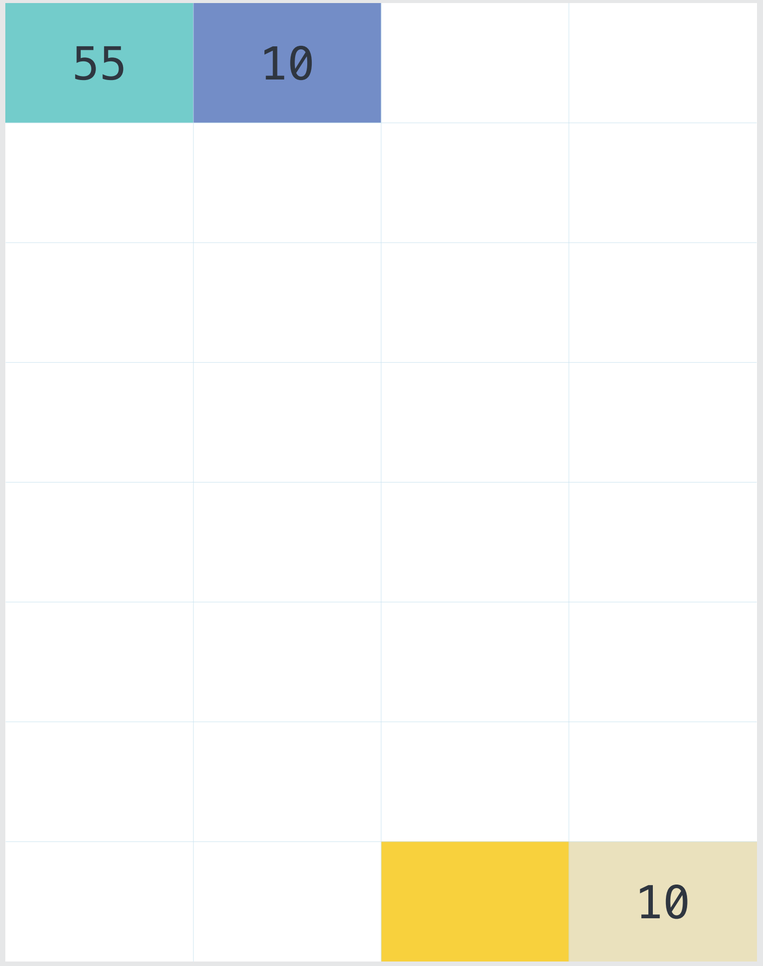
GET na vrh sklada **naloži** vsebino pomnilniške lokacije

```
000 PUSH 0
001 SET 0
002 PUSH 0
003 SET 1
004 GET 1
005 PUSH 101
006 LT
007 JMPZ 9
008 GET 0
009 GET 1
010 ADD
011 SET 0
012 GET 1
013 PUSH 1
014 ADD
015 SET 1
016 JMP -13
```



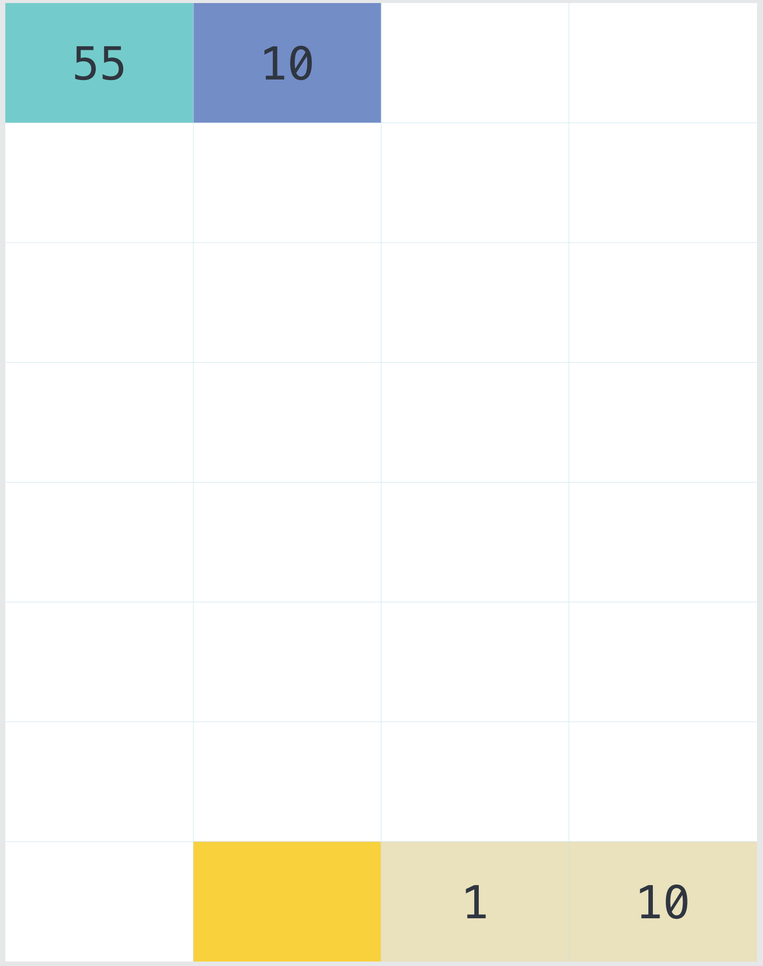
PUSH na vrh sklada doda konstanto

```
000 PUSH 0
001 SET 0
002 PUSH 0
003 SET 1
004 GET 1
005 PUSH 101
006 LT
007 JMPZ 9
008 GET 0
009 GET 1
010 ADD
011 SET 0
012 GET 1
013 PUSH 1
014 ADD
015 SET 1
016 JMP -13
```



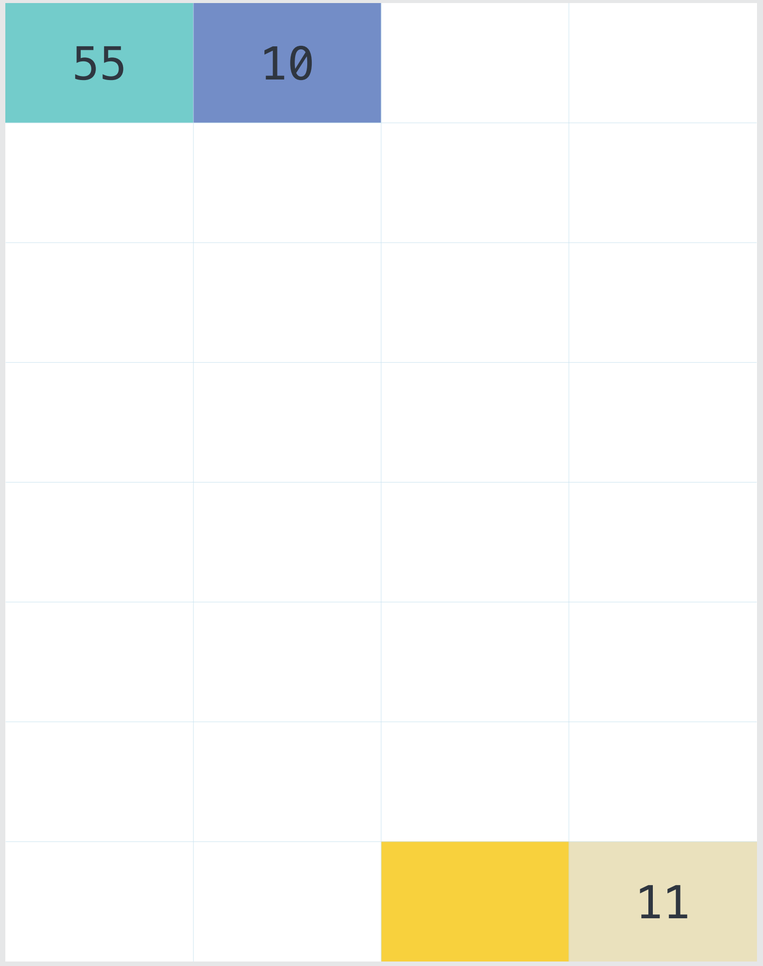
ADD vrhnji dve vrednosti nadomesti z **njuno vsoto**

```
000 PUSH 0
001 SET 0
002 PUSH 0
003 SET 1
004 GET 1
005 PUSH 101
006 LT
007 JMPZ 9
008 GET 0
009 GET 1
010 ADD
011 SET 0
012 GET 1
013 PUSH 1
014 ADD
015 SET 1
016 JMP -13
```



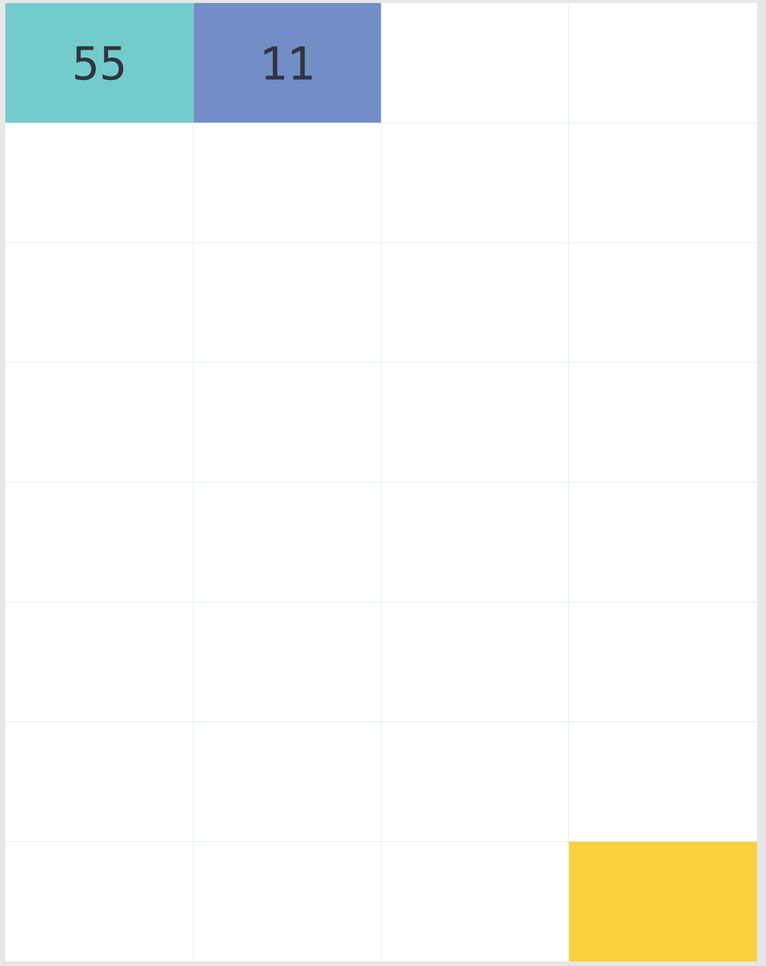
SET vrh sklada **shrani** v dano pomnilniško lokacijo

```
000 PUSH 0
001 SET 0
002 PUSH 0
003 SET 1
004 GET 1
005 PUSH 101
006 LT
007 JMPZ 9
008 GET 0
009 GET 1
010 ADD
011 SET 0
012 GET 1
013 PUSH 1
014 ADD
015 SET 1
016 JMP -13
```



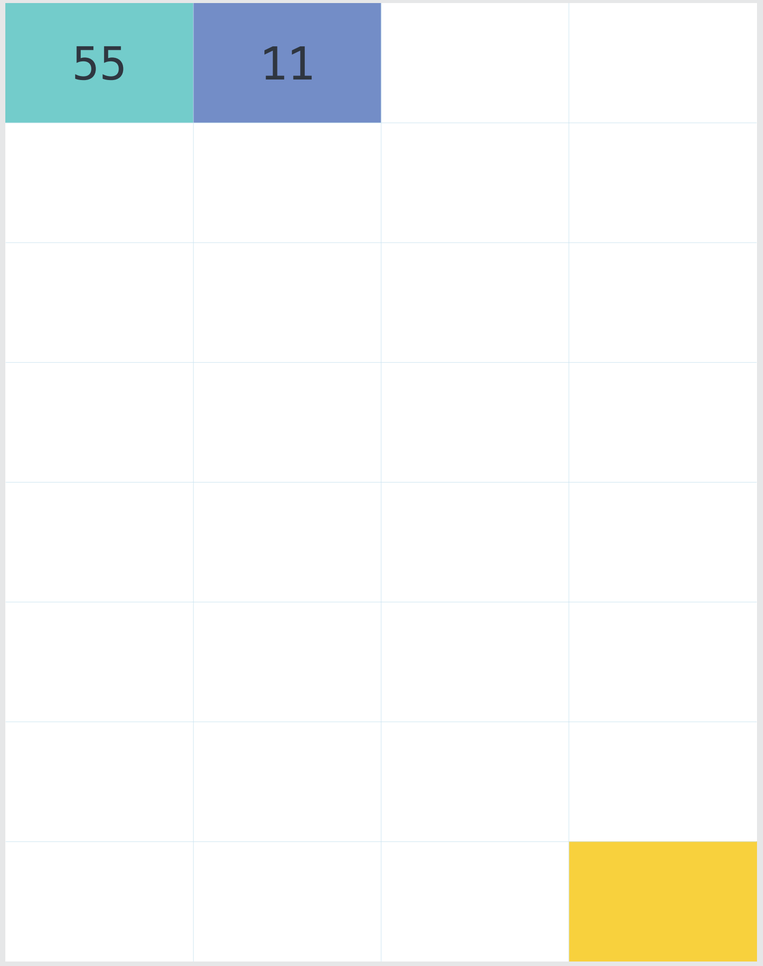
JMP naredi **brezpogojni relativni** skok

```
000 PUSH 0
001 SET 0
002 PUSH 0
003 SET 1
004 GET 1
005 PUSH 101
006 LT
007 JMPZ 9
008 GET 0
009 GET 1
010 ADD
011 SET 0
012 GET 1
013 PUSH 1
014 ADD
015 SET 1
016 JMP -13
```



GET na vrh sklada **naloži** vsebino pomnilniške lokacije

```
000 PUSH 0
001 SET 0
002 PUSH 0
003 SET 1
004 GET 1
005 PUSH 101
006 LT
007 JMPZ 9
008 GET 0
009 GET 1
010 ADD
011 SET 0
012 GET 1
013 PUSH 1
014 ADD
015 SET 1
016 JMP -13
```



virtualni stroj jezika

comm

Vsak izraz lahko **prevedemo** v zaporedje opracij

$$\begin{aligned}
\text{ukazi}(n) &= \texttt{PUSH}\, n \\
\text{ukazi}(x_i) &= \texttt{GET}\, i \\
\text{ukazi}(e_1 + e_2) &= \text{ukazi}(e_1), \text{ukazi}(e_2), \texttt{ADD} \\
\text{ukazi}(e_1 * e_2) &= \text{ukazi}(e_1), \text{ukazi}(e_2), \texttt{MUL}
\end{aligned}$$

Vsak izraz lahko **prevedemo** v zaporedje opracij

$$
\begin{aligned}
\text{ukazi}(\texttt{true}) &= \texttt{PUSH}\,1 \\
\text{ukazi}(\texttt{false}) &= \texttt{PUSH}\,0 \\
\text{ukazi}(e_1 = e_2) &= \text{ukazi}(e_1), \text{ukazi}(e_2), \texttt{EQ} \\
\text{ukazi}(e_1 < e_2) &= \text{ukazi}(e_1), \text{ukazi}(e_2), \texttt{LT} \\
&\vdots
\end{aligned}
$$

Vsak ukaz lahko **prevedemo** v zaporedje opracij

$$\begin{aligned}
\text{ukazi}(x_i := e) &= \text{ukazi}(e), \texttt{SET}\ i \\
\text{ukazi}(c_1; c_2) &= \text{ukazi}(c_1), \text{ukazi}(c_2) \\
\texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \texttt{end} &= \text{ukazi}(b), \texttt{JMPZ}\ m+1, \\
&\qquad \underbrace{\text{ukazi}(c_1)}_{m}, \texttt{JMP}\ n, \underbrace{\text{ukazi}(c_2)}_{n} \\
&\ \ \vdots
\end{aligned}$$

prevajalnik jezika

comm

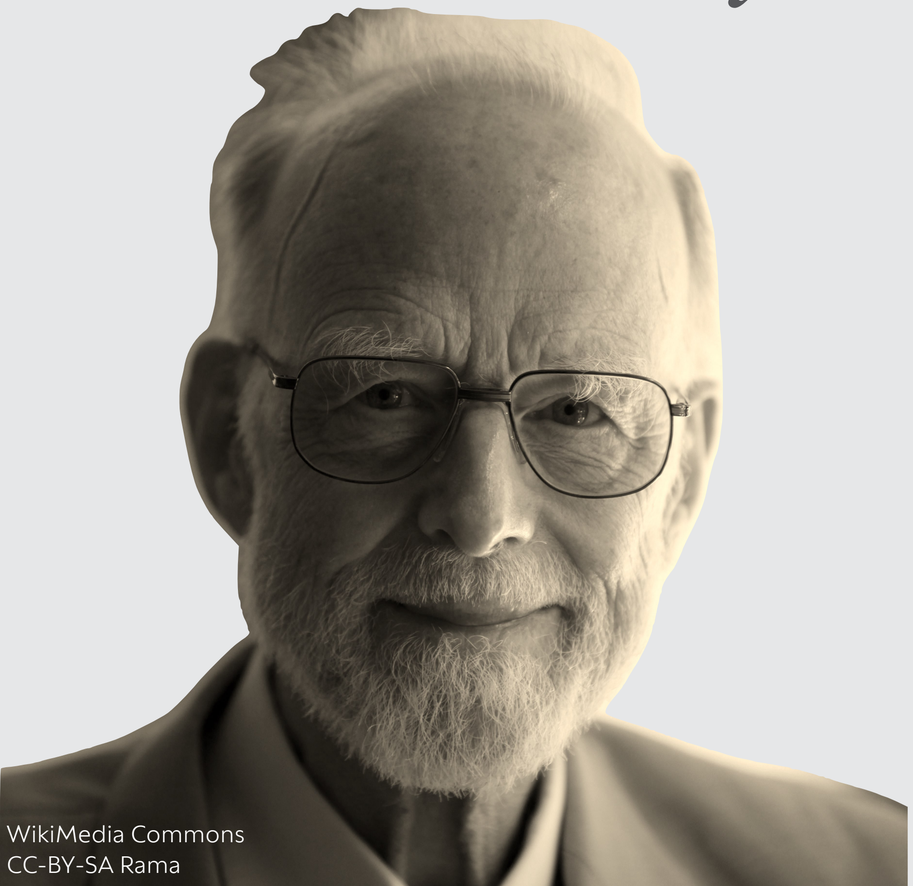
prihodnjič...

Spoznali boste **Hoareovo logiko** za dokazovanje lastnosti programov

$$\{x = m \wedge y = n\}$$

$$\texttt{if}\ x < y\ \texttt{then}\ (z := x)\ \texttt{else}\ (z := y)\ \texttt{end}$$

$$\{z = \min(m, n)\}$$



WikiMedia Commons
CC-BY-SA Rama

Charles Anthony Richard Hoare, 1934–

WikiMedia Commons
CC-BY-SA Andrej Bauer