

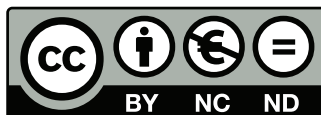
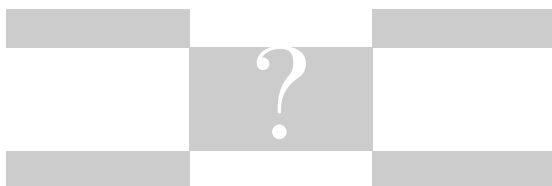
Črepinšek Matej

# ProGo

## ZAPOLNI SIVE CELICE

### Programiranje

(Draft ver. 30.11.2012)  
(Predlogi popravki na:  
[progo.crepinsek@gmail.com](mailto:progo.crepinsek@gmail.com))



Samo založba

Celje 2012

CIP – Kataložni zapis o publikaciji  
Univerzitetna knjižnica Maribor

519.854.2:004.4(075.8)

ČREPINŠEK, Matej  
ProGo ZAPOLNI SIVE CELICE Programiranje /  
Matej Črepinšek. - Celje : Zložba XXX, 2012

ISBN 86-435-0593-5  
COBISS.SI-ID 52191233

Naslov:	ZAPOLNI SIVE CELICE Programiranje
Avtor:	dr. Matej Črepinšek, univ. dipl. inž.
Vrsta publikacije:	vadnica
Strokovna recenzenta:	??
Lektorica:	?? Januš, prof. slov.
Naslovna stran:	dr. Matej Črepinšek, univ. dipl. inž.
Naklada:	1000 izvodov
Obseg:	200 strani
Izdal:	Samo založba
Natisnila:	Tiskarna Hren Ljubljana...

Creative Commons Attribution-NonCommercial-NoDerivs 2.5 Slovenia License

© Microsoft, C#, Windows in logotip Windows so zaščitene blagovne znamke ali blagovne znamke družbe Microsoft Corporation v ZDA in/ali drugih državah. © Java je zaščitena blagovna znamka ali blagovna znamka družbe Sun Microsystems, Inc. v ZDA in/ali drugih državah. © Vsi drugi izdelki, storitve, podjetja, dogodki in publikacije so blagovne znamke, zaščitene blagovne znamke ali znamke storitev svojih lastnikov v ZDA in/ali drugih državah.

# Kazalo

<b>1</b>	<b>Uvod</b>	<b>9</b>
1.1	Komu je knjiga namenjena? . . . . .	10
<b>2</b>	<b>Navodila in pojasnila</b>	<b>13</b>
2.1	Struktura in oblika . . . . .	14
<b>3</b>	<b>Proceduralno programiranje</b>	<b>15</b>
3.1	Proceduralno in deklarativno . . . . .	16
3.2	Programiranje in program . . . . .	17
<b>4</b>	<b>Izpis in konstante</b>	<b>21</b>
4.1	Določi zaporedje konstant . . . . .	23
4.1.1	Preizkusi se . . . . .	25
4.1.2	Preizkusi še težje . . . . .	26
4.2	Določi vrednosti konstant . . . . .	30
4.2.1	Pravila igre . . . . .	30
4.3	Primeri iz prakse . . . . .	36
<b>5</b>	<b>Vnos in spremenljivka</b>	<b>39</b>
5.1	Prečrtaj odvečne vrstice . . . . .	43
5.1.1	Pravila igre . . . . .	43
5.1.2	Preizkusi se . . . . .	45
5.2	Vstavi spremenljivko . . . . .	46
5.3	Primeri iz prakse . . . . .	49

<b>6</b>	<b>Vejitve</b>	<b>51</b>
6.1	Diagram poteka . . . . .	52
6.2	Primeri . . . . .	55
6.2.1	Tipske vejitve . . . . .	55
6.3	Zapiši možne poti . . . . .	56
6.3.1	Sestavljanje besed . . . . .	57
6.3.2	Izračunaj možne rezultate . . . . .	60
6.4	Poveži diagram . . . . .	62
6.4.1	Izpolnjevanje obrazcev . . . . .	64
6.5	Odločanje . . . . .	66
6.6	Zapiši izpis . . . . .	71
6.7	Izberi pogoj . . . . .	73
6.8	Poveži . . . . .	80
6.9	Primeri iz prakse . . . . .	88
<b>7</b>	<b>Ponavljanje</b>	<b>91</b>
7.1	Primeri . . . . .	93
7.2	Sled izvajanja programa . . . . .	98
7.2.1	Dopolni sled . . . . .	100
7.3	Zapiši izpise . . . . .	104
7.4	Izberi pogoj . . . . .	107
7.5	Poveži . . . . .	111
7.6	Primeri iz prakse . . . . .	116
<b>8</b>	<b>Primeri iz prakse</b>	<b>119</b>
8.1	Zahteve programa . . . . .	121
8.2	Pascal . . . . .	123
8.3	Java . . . . .	124
8.4	C# . . . . .	125
8.5	C++ . . . . .	126
8.6	Ruby . . . . .	127
8.7	Ostalo . . . . .	127
<b>9</b>	<b>Kako dalje</b>	<b>129</b>

<b>10 Priloga</b>	<b>131</b>
10.1 Izpis in konstante . . . . .	132
10.1.1 Določi vrstni red konstant . . . . .	132
10.1.2 Določi vrednosti konstant . . . . .	146
10.2 Vnos in spremenljivke . . . . .	150
10.2.1 Prečrti odvečne vrstice . . . . .	150
10.2.2 Vstavi spremenljivko . . . . .	151
10.3 Vejitve . . . . .	154
10.3.1 Sestavljanje besed . . . . .	154
10.3.2 Izračunaj možne rezultate . . . . .	156
10.3.3 Poveži . . . . .	158
10.3.4 Zapiši izpis . . . . .	162
10.3.5 Izberi pogoj . . . . .	164
10.3.6 Poveži . . . . .	170
10.4 Ponavljanje . . . . .	176
10.4.1 Zapiši izpis . . . . .	176
10.4.2 Izberi pogoj . . . . .	179
10.4.3 Poveži . . . . .	183
<b>11 Rešitve</b>	<b>189</b>



# Predgovor

Pregovor pravi: če je tvoje edino "orodje" kladivo, potem ti vsi problemi izgledajo kot "žebliji". Človek je tekom evolucije izumil že mnogo "orodij", a največje se skriva v naših možganih. Kot vsako "orodje" je tudi možgane potrebno vzdrževati s pravilno prehrano in predvsem z rednimi miselnimi izzivi. Pred vami je knjiga, ki vas ob različnih miselnih igrah spodbudi k logičnemu proceduralnemu načinu mišljenja, ob tem pa še dodatno spoznate osnove računalniškega programiranja.

Programiranje je eno izmed najrazvitejših načinov podajanja proceduralnega znanja in hkrati skrito skoraj vsem, ki se ne ukvarjajo z računalniškim programiranjem. Osnove iz računalniškega programiranja vam ne bodo pomagale samo pri lažjem razumevanju računalniških programov, ampak jih lahko uporabite tudi na drugih področjih.

Programiranje predstavimo brez uporabe računalnika, pri tem pa si pomagamo z najrazličnejšimi igrami in tehnikami iz časov, ko so programe pisali in preverjali še na papir (diagrami poteka, programske sledi, itd.). Knjiga je napisana tako, da je primerna široki množici bralcev, ki ne potrebujejo posebnega predznanja.

Celje, oktober 2012

Matej Črepinšek

*progo.crepinsek@gmail.com*

---



# 1.

*Moji možgani - so moj drugi najljubši organ.*

*- Sleeper (film), Woody Allen -*

## Uvod

### Vprašanja

- ★ Želite se igrati?
- ★ Želite ohranjati mentalno kondicijo?
- ★ Želite razširiti svoje znanje?
- ★ Želite izzvati svoje možgane in si "razgibati" sive celice?
- ★ Želite spoznati osnove programiranja?
- ★ Želite izboljšati razumevanje delovanja elektronskih naprav?
- ★ Ne želite polniti možgane z velikimi količinami nepomembnih podatkov?
- ★ Želite razumeti in biti razumljen?

Seveda vseh teh želja ta knjiga ne more izpolnit, lahko pa je majhen kamenček v mozaiku, ki vodi do njihove izpolnitve.

Vsak izmed nas ima edinstven prstni odtis, še veliko bolj edinstveni pa so naši možgani. Neskončne možganske kapacitete čakajo, da jih odkrijemo in uporabimo. Za mnoge je programiranje, in z njim povezano proceduralno znanje, ena izmed manj znanih področij znanja in kot takšno primerno za razgibanje možganskih celic. Da bi omogočili čim lažji in zanimivejši vpogled v svet programiranja smo pri-

pravili najrazličnejše primere in igre, katerih cilj je spoznati in preizkusiti osnovne koncepte programiranja. Znan pregovor pravi, da vaja dela mojstra in pred vami je knjiga z več kot 270 vajami. Vsaka izmed vaj zahteva različno stopnjo miselnega napora.

## 1.1 Komu je knjiga namenjena?

Knjiga je napisana tako, da je primerna kar najširšemu krogu ljudi. Ker knjiga govori o programiranju, bi bralec lahko pomislil da potrebuje računalnik, vendar v našem primeru ni tako. Za uspešno spoznavanje programiranja potrebujete samo pisalo. Svetujemo, da podobno kot pri reševanju križank, uporabljate svinčnik in radirko. Knjigo posebej priporočamo naslednjim skupinam:

- Vsem, ki bi radi vzdrževali in povečevali možgansko kondicijo. Reševanje križank predvsem spodbuja centre v možganih, ki so povezani s spominom. Programiranje oz. igre v knjigi pa spodbujajo centre v možganih, ki so povezani s proceduralnim in logičnim mišljenjem.
- Vsem, ki radi rešujejo logične igre, kot je Soduko.
- Vsem, ki se želijo naučiti programiranja. Knjigo se lahko uporabi za pripravo, ali prvi korak v svet programiranja. Razumevanje različnih knjig o programskih jezikih, tj. jezikih v katerih pišemo računalniške programe, bo po tej knjigi lažje, saj bo bralec seznanjen z osnovnimi miselnimi procesi, potrebnimi za razumevanje programiranja.
- Vsem, ki si želijo poglobiti znanje o delovanju in uporabi različne programske opreme, saj programske jezike

srečamo kot napredno funkcionalnost v preglednicah, urejevalnikih besedil, predstavitvenih programih, itd.

- Vsem osnovnošolcem, ki obiskujejo krožke iz računalništva.
- Vsem dijakom in študentom, ki bi radi izboljšali svoje sposobnosti.
- Vsem dijakom, ki se odločajo za študij na tehniških fakultetah.

Zaradi splošnosti bo za mnoge veliko iger preprostih in s tem mogoče premalo zanimivih, kljub temu pa ne smemo pozabiti, da za osvojitev posameznih vzorcev razmišljanja potrebujemo veliko različnih ponovitev. Za izboljšanje kondicije sta potrebni kvaliteta in kvantiteta.

*progo.crepinsek@gmail.com*

---

## 2.

*Če nič ne dela, preberi navodila.*

*- nasvet iz svetovnega spleta -*

# Navodila in pojasnila

### Znano

- ★ Navodila se redko berejo.
- ★ Od navodil je pogosto odvisna naša uspešnost.
- ★ Pisanje navodil je zahtevno.

Cilj poglavja je podati bralcu osnovne informacije o uporabi knjige.

Knjiga zahteva od bralca, da se prelevi v igralca in aktivno sodeluje, saj ga čakajo različne logično proceduralne igre.

Za uspešno sodelovanje potrebuje bralec dobro voljo, **svinčnik** in **radirko**. Rezultati se pišejo v sive celice (sivi okvirčki).



Vse igre so zasnovane po principu detektivke, kjer imate podane vse podatke, da razrešite skrivnost. V praksi to pomeni, da so podani začetni podatki (vhod) in pričakovan rezultat (izhod), na vas pa je, da jih pravilno povežete, določite ali prečrtate.

## 2.1 Struktura in oblika

Vsako poglavje se začne z okvirčkom "Znano". Namen dejstev je, da bralca opozori na že znane podatke in mu omogoča lažje povezovanje znanja z že znanim.

Sledi opis in razlaga novih programskih ukazov (razen uvodnih poglavij) Ukazi se skozi poglavja nadgrajujejo in dopolnjujejo, zato preskakovanje celotnih poglavij odsvetujemo.

Po opisu osnov, sledijo različne igre, ki omogočajo utrditev in poglobitev pridobljenega znanja. Pred vsako novo igro so navodila in primer rešene naloge.

Temu sledijo naloge. **Vsaka naloga je označena s trenutno stranjo in zaporedno črko.** Rešitev za vsako nalogo je podana proti koncu knjige, v poglavju "Rešitve". Naloge so za lažje določanje uspešnosti, označene s težavnostjo. Težavnost posamezne naloge je označena s številom zvezdic. Najlažje naloge so označene z eno zvezdico, tj. oznako  in najtežje naloge s petimi zvezdicami, tj. oznako .

Vsaka tema vsebuje več različnih nalog. Bralcu odsvetujemo ob prvem branju rešiti vse naloge, ampak je dovolj da jih reši pri vsaki igri samo nekaj. Ostale naloge pa si pusti za kasnejše ponavljanje. Vsaka igra ima še dodatne naloge, ki jih najdete v prilogi.

Ob koncu vsakega poglavja je dodano še **informativno poglavje** z naslovom "Primeri iz prakse". Cilj poglavja je predstaviti spoznane ukaze na različnih programskih jezikih. **Razumevanje poglavja "Primeri iz prakse" ni pomembno za uspešno razumevanje nadaljnjih poglavij.**

# 3.

*Kdor hitro da, dvakrat da.*

*- Latinski pregovor -*

## Proceduralno programiranje

Svet računalništva in s tem tudi pristopi programiranja računalnikov se naglo razvijajo. Danes poznamo različne vrste programiranja od proceduralnega, dokotkovnega do funkcijskega. V knjigi se omejimo na proceduralno programiranje, ki je osnova za razumevanje delovanja računalniških programov in je bližje računalniškemu jeziku.

## 3.1 Proceduralno in deklarativno

### **Znano**

- ★ Znanje lahko delimo na deklarativno in proceduralno.
- ★ Vedeti, da je Ljubljana glavno mesto Slovenije, je primer deklarativnega znanja.
- ★ Znati voziti avto, je primer proceduralnega znanja.
- ★ Procedura je postopek.
- ★ Deklaracija ja lahko javna izjava.
- ★ Vedeti nogometna pravila je primer deklarativnega znanja.
- ★ Igrati nogomet je primer proceduralnega znanja.

Pridobljeno znanje lahko delimo na proceduralno in deklarativno. Za pridobivanje proceduralnega znanja je potrebna vaja. Veljalo je, da se mnogokrat v šolah pridobi več deklarativnega, kot proceduralnega znanja, danes pa je vedno več poudarka na praktični uporabi znanj. Učenje zgodovinskih in zemljepisnih podatkov, matematičnih enačb, kemičnih lastnosti, itd., mnogokrat praktično ne znamo uporabiti, s tem pa nas takšno znanje ne spodbuja k nadaljnjemu raziskovanju. Uspeh na praktičnem primeru nam nudi ne samo znanje ampak tudi notranje zadovoljstvo. Kot primere takšnega zadovoljstva lahko štejemo: rešeno logično igro, rešeno križanko, odigrano računalniško igro, pečeno sladico, sestavljeno omaro itd., skratka vsako uspešno opravljeno delo. Tako celotno spoznavanje temelji na kar največjem različnih iger, ki jih utrjujemo s pomočjo vaj.

Z razvojem strojništva in računalništva se je povečevala potreba po kvalitetnem podajanju navodil, npr. v primeru računalništva je to programiranje. Programiranje velja za zelo natančen in dobro definiran način podajanja nalog. V



knjigi želimo predstaviti podajanje znanja s pomočjo idej in izkušenj pridobljenih pri računalniškem programiranju.

## 3.2 Programiranje in program

### **Znano**

- ★ Program je skupek nalog, del, ki se določijo za uresničitev zadanega cilja.
- ★ Nastavitev pralnega stroja na program za hitro pranje. Program so izdelali izdelovalci pralnega stroja.
- ★ Pojem "programiranje vedno snemalnika" pomeni, da nastavimo televizijske programe, samodejno snemanje najljubše oddaje, itd...
- ★ Programiranje telefonske tajnice je postopek nastavljanja različnih lastnosti odzivanja telefonskega aparata. Telefon se po postopku programiranja vedno odzove po določenem postopku.

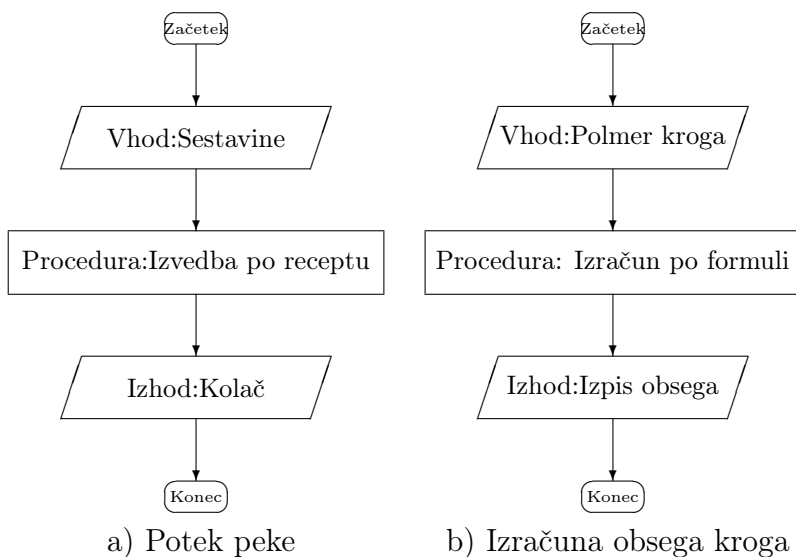
Program in programiranje v takšnih ali drugačnih oblikah, danes srečamo skoraj že na vsakem koraku. Večino teh programov so za nas pripravili proizvajalci elektronskih naprav, kadar pa želimo da se elektronske naprave odzivajo na nam prirejen način pa smo tudi sami vede ali nevede vpleteni v proces programiranja.

Programiranje bi lahko definirali kot proces, katerega cilj je program. Program pa predstavlja zaporedje navodil, ki so potrebni za uresničitev neke, določene naloge. Najenostavnejša navodila imenujemo ukazi.

Najrazvitejše je programiranje na področju računalništva. Zaradi kompleksnosti in možnih nejasnosti naravnih jezikov (Slovenščina, Angleščina, itd...), so na področju računal-

ništva razvili ti. programske jezike. V praksi srečamo programske jezike kot so: *Java*, *C#*, *Pascal*, *Ruby* in drugi. Kljub temu, da ima večina programskih jezikov svoje besede in pravila (sintakso) različna, imajo večinoma podoben način podajanja pravil (koncepte).

Program se običajno sestoji iz trojice: vhod, osrednji del in izhod.



Slika 3.1: Primera diagrama poteka za peko kolača in izračin obsega kroga

Vhod določa podatke (ali surovine) potrebne za izvedbo opravila (programa). Osrednji del opisuje dejanja, ki so potrebna za pridobitev željenega rezultata. Izpis je namenjen obveščanju ali vračanju rezultata. Kot vsakdanja primera definiranih vhodov, izhodov in osrednjega dela vzemimo primera poteka peke kolača in izračuna obsega kroga (slika 3.1). Na sliki 3.1 je potrebno še poudariti, da je vrstni red izvoja-

nja pomemben (vrstni red opisujejo smeri puščic, ki povezujejo simbole diagrama).

*progo.crepinsek@gmail.com*

---

# 4.

*Najlepša in najtežja sta začetek in konec!*

## Izpis in konstante

### Znana dejstva

- ★  $\pi = 3,14159\dots$  - znana matematična konstanta.
- ★ Konstanta je količina, ki ne spreminja svoje vrednosti.
- ★ Konstanta je sestavljena iz imena in vrednosti.
- ★ Vse kar se prikaže na ekranu računalnika je posledica **izpisa** programov.

Ena izmed pomembnejših nalog programiranja je izpis podatkov. V praksi to v večini primerov pomeni prikaz podatkov na zaslonu računalnika, telefona, ali kakšne druge naprave. Prikaz na zaslon bomo definirali z ukazom **izpis**, kateremu sledi besedilo.

Primer ukaza	Izpis (na ekran)
<code>izpis "Danes je lep dan!"</code>	Danes je lep dan!
<code>izpis "abrakadabra"</code>	abrakadabra
<code>:</code>	:

Kadar želimo enako besedilo ali vrednost izpisati ali uporabiti večkrat, ga označimo z imenom, tj. imenom konstante. Konstanto torej določa ime in vrednosti, ki je lahko besedilo ali številčna vrednost. Npr. če za besedilo "Zal danes nimam časa!" uporabimo oznako **x** in za definiranje znak **=** (za lažjo

ločitev med oznakami in besedilom, zapišemo besedilo med narekovaja) dobimo:

```
x = "Žal danes nimam časa!"
```

Sedaj velja, da v programu `x` nadomesti besedilo `Žal danes nimam časa!`

Zapišimo nekaj primerov ukazov:

Primer ukaza	Izpis
<code>x = "Žal danes nimam časa!"</code>	
<code>izpis "Greš v kino?"</code>	Greš v kino?
<code>izpis x</code>	Žal danes nimam časa!
<code>izpis "Gremo karte?"</code>	Gremo karte?
<code>izpis x</code>	Žal danes nimam časa!
<code>:</code>	<code>:</code>

Sledi opis logične igre "Določi zaporedje konstant".

## 4.1 Določi zaporedje konstant

Cilj igre je določiti pravo zaporedje konstant (dogodkov), ki bo privedlo do določenega izpisa. Pri igri so vnaprej podane možne konstante in izpis.

V spodnjem primeru določimo konstanti  $x$  in  $y$ :

$x = \text{"bc"}$

$y = \text{"aaa"}$

Iščemo izpis `bcaaaaaabcbc`. Torej iščimo zaporedje konstant  $x$  in  $y$ , ki izpiše besedilo `bcaaaaaabcbc`, ali zapisano drugače, kako razdeliti besedilo `bcaaaaaabcbc`, da ga lahko zapišemo kot zaporedje konstant  $x$  in  $y$ .

Igra je podana v obliki tabele, kjer moramo v sivo celico (kvadrat), vpisati zaporedje konstant, ki opisuje niz, zapisan nad dvojno črto.

	bcaaaaaabcbc				
	<hr/> <hr/>				
izpis	? ? ? ? ...				

Rešitev zgornje igre je zaporedje:  $x$ ,  $y$ ,  $y$ ,  $x$  in  $x$ , tj.:

	bc	aaa	aaa	bc	bc
	<hr/> <hr/>				
izpis	x	y	y	x	x

Za lažje reševanje smo v nadaljevanju vse znake izpisov opremili z zaporednimi števkami.

**Namig:** Svetujemo, da uporabnik najprej s svinčnikom označi mesta kjer se niz deli in če se na koncu izide, zapiše pripadajoče zaporedje konstant.

**Namig:** Če se razreševanje niza od leve proti desni zaustavi, svetujemo drug pristop, na primer razreševanje iz desne proti levi.



### 4.1.1 Preizkusi se

V sive celice (okvirčke) zapišite pravilni vrstni red konstant x, y, z in w.

\* Zvezdice na desni strani posamezne igre, predstavljajo zahtevnost igre. Rešitve lahko preverite v poglavju "Rešitve". Rešitev naloge opisuje stran in oznaka naloge (zaporedna črka).

```
x = "bc"
y = "aaa"
z = "aabc"
w = "aabb"
```

- a) 

	1	2	3	4	5	6	7	8	9	10	11	12	
	bcaaaaaabcabc												★
izpis	<div></div>												
- b) 

	1	2	3	4	5	6	7	8	9	10	11	12	
	bcaaaaaaaabc												★
izpis	<div></div>												
- c) 

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
	aabbbaabbaabc														★★
izpis	<div></div>														
- č) 

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
	aabbaabcaaaaabb															★★
izpis	<div></div>															
- d) 

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
	aabbaabbbaaaaaaabb																		★★★
izpis	<div></div>																		
- e) 

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
	bcaabcaabbaabcaabc																		★★★
izpis	<div></div>																		

### 4.1.2 Preizkusi še težje

Osnovna pravila igre ostajajo ista. Razlika je v tem, da dobljeno zaporedje konstant, ne predstavlja več končno rešitev, ampak iskani niz v drugem koraku. Število korakov, do končne rešitve opisuje število sivih celic. Da je reševanje zanimivejše sta lahko določena zadnji in vmesni izpis.

Npr. vzemimo primer, kjer iščemo dva vmesna koraka. Zadnji korak je že določen. V vsakem koraku iščemo zaporedje konstant  $a$ ,  $b$  in  $c$ .

	ababcbabccacaab			
$a = "ab"$	izpis <sub>1</sub>	? ? ? ?		
$b = "abc"$	izpis <sub>2</sub>	? ?		
$c = "ca"$	izpis <sub>3</sub>	b		

Iščemo pravilne izpeljave za  $izpis_1$  in  $izpis_2$ , tako da velja  $izpis_3$ . Rešitev je:




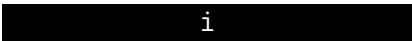
	ab abc ab abc ca ca ab			
	izpis <sub>1</sub>	ab abc ca		
	izpis <sub>2</sub>	abc		
	izpis <sub>3</sub>	b		

**Namig:** Ko se razreševanje niza od zgoraj navzdol zaustavi zaradi različnih možnih rešitev, si pomagajte z razreševanjem od spodaj navzgor.

Določi zaporedje konstant **a**, **b**, **i** in **r**.




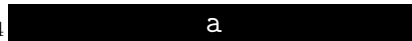
```
a = "ri"
b = "bi"
i = "ra"
r = "ba"
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16  
birabarabiribari

a) izpis<sub>1</sub>   
izpis<sub>2</sub>   
izpis<sub>3</sub>   
izpis<sub>4</sub>  i




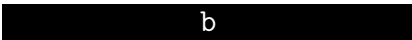
★★

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16  
birabarabiribara

b) izpis<sub>1</sub>   
izpis<sub>2</sub>   
izpis<sub>3</sub>   
izpis<sub>4</sub>  a

★★

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16  
birabaribiribara

c) izpis<sub>1</sub>   
izpis<sub>2</sub>   
izpis<sub>3</sub>   
izpis<sub>4</sub>  b

★★

Določi zaporedje konstant `a`, `v` in `j`.

Igra je razširjena s pravilom, da lahko ena konstanta opisuje več različnih tekstov. V našem primeru konstanta `a` opisuje `"ja"` ali `"av"`.

`a = "ja"` ali `a = "av"`

`v = "aj"`

`j = "va"`

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16  
avvavajaajjavaav


a) izpis<sub>1</sub> 

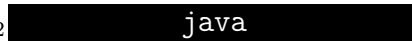

izpis<sub>2</sub>  


izpis<sub>3</sub> 


izpis<sub>4</sub> 

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16  
ajjaavajavvavaav

b) izpis<sub>1</sub> 


izpis<sub>2</sub>  



izpis<sub>3</sub> 


izpis<sub>4</sub> 


*\*java je ime za programski jezik.*

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16  
ajavvaavvajaajja

c) izpis<sub>1</sub> 

izpis<sub>2</sub>  

izpis<sub>3</sub> 

izpis<sub>4</sub> 

Določi zaporedje konstant a, b in c.

a = "bc" ali a = "caba"

b = "ca"

c = "ba"

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
	babcbacabacabc														
a) izpis <sub>1</sub>	<div></div>														★★★
izpis <sub>2</sub>	<div></div>														
izpis <sub>3</sub>	<div>c</div>														

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
	cabccababacabacabc																		
b) izpis <sub>1</sub>	<div></div>																		★★★★
izpis <sub>2</sub>	<div></div>																		
izpis <sub>3</sub>	<div></div>																		
izpis <sub>4</sub>	<div>c</div>																		

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
	cabcbabccabcbabccaba																				
c) izpis <sub>1</sub>	<div></div>																				★★★★
izpis <sub>2</sub>	<div></div>																				
izpis <sub>3</sub>	<div></div>																				
izpis <sub>4</sub>	<div>a</div>																				

## 4.2 Določi vrednosti konstant

Tokrat za razliko od prejšnje igre iščemo vrednosti konstant, medtem ko je zaporedje izpisa znano.

### 4.2.1 Pravila igre

Cilj igre je določiti vrednosti konstant tako, da ustrezajo izpisu. Za začetek si poglejmo nekaj najenostavnejših primerov.

Določi  $x =$   ? pri izpisu:

	ab
<hr/>	
izpis	<input type="text"/> x

Za pravilno rešitev mora  $x$  izpsat **ab**, zato je rešitev  $x = \text{"ab"}$ .

Določi  $x =$   ? pri izpisu:

	abab
<hr/>	
izpis	<input type="text"/> x x

Ukaz izpis je sestavljen iz dveh konstant ( $x$   $x$ ), zato je potrebno deliti **"abab"** na dva dela in sicer tako, da bosta enaka (ker sta obe konstanti enaki, tj.  $x$ ). V sivo celico zapišemo kot rešitev  $x = \text{"ab"}$ .

**Namig:** Bodite pozorni na dolžino posameznih konstant, tj. koliko znakov opisuje ena konstanta.

Poglejmo si primer, kjer iščemo vrednosti konstant  $x$  in  $y$ .

$x =$  ?    pri izpisu  $\frac{\text{abaa}}{\text{izpis } \text{x y x}}$   
 $y =$  ?

Niz "abaa" je potrebno deliti na tri dele ( $x y x$ ), in sicer tako da bosta začetek in konec enaka (tam je  $x$ ). Če poizkusimo za  $x="ab"$ , vidimo da se nam konec ne izide, ker sta na koncu niza znaka "aa". Začetek in konec sta enaka samo pri znaku "a". Zato je  $x="a"$ . V osrednjem delu nam ostane "ba", zato je  $y="ba"$ .

$x =$  "a"  
 $y =$  "ba"

Za zahtevnejši primer iščemo vrednosti konstant  $x$  in  $y$ , pri izpisu:

$\frac{\rightarrow \text{bcaaaaaabcb} \leftarrow}{\text{izpis } \text{x y y x x}}$

Pri reševanju si pomagamo s sklepanjem iz izpisa. V našem primeru se izpis začne s črko "b" in ukaz izpis s konstanto  $x$ . Torej vrednost  $x$ -a se začne z "b". Nato lahko sledi črka "c". Ker je  $x$ , tudi zadna konstanta v izpisu in izpis se konča s črko "c", lahko sklepamo, da je vrednost  $x$ -a enaka "bc". Če sedaj v izpisu označimo vse izpisane "bc", nam ostane "aaaaaa" in konstanti  $yy$ . Sklepamo, da je vrednost konstante  $y$  enaka "aa".

$x =$  "bc"  
 $y =$  "aa"

Število različnih konstant in njuna imena nam razkriva tabela, ki jo mora igralec izpolnit.

Določi vrednost konstantam.

a)  $x =$

★

$\begin{array}{cccc} 1 & 2 & 3 & 4 \\ & c & d & c & d \end{array}$   
=====  
izpis

b)  $x =$    
 $y =$

★★

$\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \\ & a & a & b & b & a & a \end{array}$   
=====  
izpis

c)  $x =$    
 $y =$

★★

$\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \\ & a & b & a & b & b & a \end{array}$   
=====  
izpis

č)  $x =$    
 $y =$

★★★

$\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \\ & a & a & b & b & a & a \end{array}$   
=====  
izpis



Določi vrednosti konstant **x**, **y** in **z**. Za lažje določanje vrednosti, sta določena dva izpisa.

x =   
 a) y =   
 z =

★★★

	1	2	3	4	5	6	7	8	9	10
	babaacacab									
<hr/>										
izpis	y y z z x									
	1	2	3	4	5	6	7	8	9	10
	acbaacbaab									
<hr/>										
izpis	z y z y x									

x =   
 b) y =   
 z =

★★★

	1	2	3	4	5	6	7	8	9	10
	ddddadaaaa									
<hr/>										
izpis	x x y z z									
	1	2	3	4	5	6	7	8		
	adddaadd									
<hr/>										
izpis	y x z x									

Določi vrednosti konstant  $x$ ,  $y$  in  $z$ .

$x =$    
 a)  $y =$    
 $z =$

★★★

	1	2	3	4	5	6	7	8	9	
	a	b	a	b	a	a	a	a	b	
<hr/>										
izpis	z z x z									
<hr/>										
	1	2	3	4	5	6	7	8	9	10
	a	b	a	a	a	a	a	a	a	a
<hr/>										
izpis	z x y x									

$x =$    
 b)  $y =$    
 $z =$

★★★★

	1	2	3	4	5	6	7	8				
	0	0	0	1	1	0	0	0				
<hr/>												
izpis	x y z x											
<hr/>												
	1	2	3	4	5	6	7	8	9	10	11	12
	1	0	0	1	0	0	0	0	0	1	0	0
<hr/>												
izpis	z y x x y x											

Določi vrednosti konstant  $x$ ,  $y$ ,  $z$  in  $w$ .

a)

$x =$	<input type="text"/>	★★★★
$y =$	<input type="text"/>	
$z =$	<input type="text"/>	
$w =$	<input type="text"/>	

```

      1 2 3 4 5 6 7 8 9 10 11
      10110110110
=====
izpis  z x x y z
      1 2 3 4 5 6 7 8 9 10 11 12
      001010011010
=====
izpis  w z y w x z

```

b)

$x =$	<input type="text"/>	★★★★
$y =$	<input type="text"/>	
$z =$	<input type="text"/>	
$w =$	<input type="text"/>	

```

      1 2 3 4 5 6 7 8
      aabaaaab
=====
izpis  z w z x y
      1 2 3 4 5 6
      aaabba
=====
izpis  x z y w
      1 2 3 4 5 6 7 8
      bababaaa
=====
izpis  w w y x z

```

Več nalog različnih težavnostnih stopenj lahko najdete v prilogi na straneh od 132 do 150.

## 4.3 Primeri iz prakse

Poglavje "Primeri iz prakse" srečamo ob koncu vsakega poglavja. Poglavja so informativne narave in poznavanje le teh ni obvezno za nadaljnje razumevanje. Celotni primeri programov, v različnih programskih jezikih, so podani v poglavju 8, ki se začne na strani 119.

Poglejmo si kako definiramo in izpišemo konstanto v različnih programskih jezikih. Definirali in izpisali bomo konstanto z imenom `ime1` in vrednostjo `abc`, tj. v našem zapisu ukaza:

**`ime1 = "abc"`**

**izpis `ime1`**

Programski jezik	Ukaz
Pascal	<b><code>const ime1: string = 'abc'; writeln(ime1);</code></b>
Java	<b><code>static final String IME1 = "abc"; System.out.println(IME1);</code></b>
C#	<b><code>const string ime1 = "abc"; Console.WriteLine(ime1);</code></b>
C++	<b><code>const char* ime1 = "abc"; cout&lt;&lt; ime1;</code></b>
Ruby	<b><code>Ime1 = 'abc' puts Ime1</code></b>

V tabeli so besede, ki so del programskega jezika ali

njegovih knjižnic, zapisane s krepkimi črkami. Če sedaj primerjamo zapise v različnih programskih jezikih opazimo, da se za isti ukaz uporabljajo različne besede, kljub temu pa je razvidno kateri del predstavlja ime konstante in kateri vrednost. Splošno velja, da so si osnovne lastnosti v različnih programskih jezikih definirane podobno.

Konstanto ponavadi definira na začetku programa in ustrezno komentira. Ko je vrednost konstante definirana, velja za celoten program enaka vrednost.

Kadar želimo v programu spreminjati vrednost uporabimo namesto konstante spremenljivko, kar je tema naslednjega poglavja.

*progo.crepinsek@gmail.com*

---

# 5.

*Ne skrbi, če potuješ počasi; skrbi  
naj te, ko obstaneš na mestu.*

*- Kitajski pregovor -*

## Vnos in spremenljivka

### Znano

★ Matematiki uporabljajo namesto izraza spremenljivka, izraz neznanka.

★  $y = k \cdot x + n$  je splošna oblika linearne funkcije, ki povezuje spremenljivki  $x$  in  $y$ .  $k$  in  $n$  sta konstanti.

★ Obrestna mera je spremenljivka, ki se spreminja glede na inflacijo.

★ Rezultat na košarkarski tekmi je definiran s spremenljivkama "domači" in "gosti". Rezultat na semaforju je prikaz-izpis vrednosti spremenljivk.

V predhodnjem poglavju smo spoznali ukaz **izpis**, ki ga potrebujemo za prikaz podatkov (npr. na ekran). Brez izpisa so v računalniku samo trenutna stanja elektronskih elementov, ki niso človeško berljiva. V primeru ko program sestavljajo samo konstante in izpis, velja da računalnik vedno vrne isti rezultat. Uporaba takšnega programa omeji uporabo računalnika na prikazovalnik vnaprej določenega besedila. Uporabnost razširimo tako, da tekom izvajanja programa omogočimo vnos podatkov, ki nato vplivajo na izvajanje programa

in izpis. Vnesene podatke si moramo nekako zapomniti, da se lahko nato v programu sklicujemo nanje. Vse navedeno nam omogočajo prav **spremenljivke**.

Za lažje razumevanje si pogledjmo primer osnovnošolske naloge iz matematike, ki zahteva izračun ploščine pravokotnika. Za izračun je podana formula:

$$ploscina = a \cdot b$$

**Primer naloge:** Izračunaj ploščino pravokotnika, ki ima stranici  $a = 5$  in  $b = 3$ .

Za rešitev naloge vstavimo števili v formulo ( $ploscina = 5 \cdot 3$ ) in in dobimo rezultat (15).

V primeru dolžine stranic  $a = 4$  in  $b = 2$ , dobimo s pomočjo vstavljanja števil v formulo in poštevanka rezultat 8.

V programu nalogo rešimo tako, da sta stranici  $a$  in  $b$  spremenljivki, ki ju vnese uporabnik, formula pa *ukaz* v obliki **izraza**.

Sedaj definiramo še ukaz **vnos**. Ko program izvaja ukaz **vnos**, se ustavi in čaka uporabnika, da vnese podatek. Prebrana vrednost se zapiše v spremenljivko, ki sledi ukazu **vnos**. Primera vnosa za zgornjo nalogo sta:

```
vnos a    // program čaka, da uporabnik vnese vrednost
vnos b    // program čaka, da uporabnik vnese vrednost
```

Dodajmo še spremenljivko *ploscina*, ki je ne vnašamo, ampak izračunamo s pomočjo izraza.

```
ploscina = a * b // rezultat se zapiše v spremenljivko
```

Večina programskih jezikov ima vgrajeno možnost računanja aritmetičnih izrazov, tj. seštevanje, odštevanje, množenje itd. Za razliko od žepnega računalnika, kjer vnašamo samo vrednosti in operatorje, lahko sedaj vnašamo namesto števil tudi aritmetične izraze (formule). Zaradi omejitev prvih



prikazovalnikov, se v aritmetičnih izrazih namesto matematičnega znaka za množenje '.' uporablja znak '\*'. Podobno velja za deljenje, kjer dvopičje zamenja znak '/', namesto ulomkove črte pa uporabimo oklepaje. Ulomkovo črto (npr.  $\frac{a+1}{2}$ ) pa nadomestimo s pravilno postavljenim '/' in oklepaji (dobimo (a+1)/2).

Izračuni se opravljajo tekom izvajanja programa. V primeru izvajanja ukaza `ploscina = a * b` se rezultat (izračun desne strani) zapiše v spremenljivko `ploscina`, ki je na levi strani enačaja, v primeru ukaza `izpis a * b` pa se vrednost izpiše na ekran.

Zapišimo nekaj primerov ukazov:

	Primer ukaza	Izpis
1	<code>x = 1 + 4 / 2</code>	
2	<code>izpis "Rezultat je:" x</code>	Rezultat je: 3
3	<code>x = 99</code>	
4	<code>izpis "Rezultat je:" x</code>	Rezultat je: 99
5	<code>y = (x + 1) / 2</code>	
6	<code>izpis "Rezultat je:" y</code>	Rezultat je: 50
:	:	:

Ukaze programa zapišemo v takšnem vrstnem redu, kot se program izvaja, tj. zaporedno od zgoraj navzdol. Zapišimo še sedaj celotni program za izračun ploščine pravokotnika:

1. `vnos a` // uporabnik vnese število, npr. 3
2. `vnos b` // uporabnik vnese število, npr. 4
3. `ploscina = a * b`
4. `izpis ploscina` // program izpiše število 12

*V programih pogosto zapišemo še dodatne informacije, ki pojasnjujejo posamezne dele programa. Takšnim opisom pravimo komentariji. V našem primeru smo uporabili vrstične*

*komentarje, za katere velja, da se nahajajo desno od ukazov, ločuje jih oznaka '//'.*

Navedeni program je do uporabnika neprijazen, saj ne izpiše nobenih navodil, kaj naj uporabnik vnese in kaj pomeni rezultat.

Program lahko zapišemo bolje:

```
1. izpis "Program za izračun ploščine pravokotnika"
2. izpis "Vnesite stranico a:"
3. vnos a
4. izpis "Vnesite stranico b:"
5. vnos b
6. ploscina = a * b
7. izpis "Ploščina je:" ploscina
```

Kot smo že omenili se pri izračunih najprej izračuna desna stran, ki se nato izpiše ali priredi spremenljivki. Poglejmo si primer, kjer ista spremenljivka nastopi na levi in desni strani računa.

```
1. x = 0
2. x = x + 2
3. x = x + 2
4. x = x + 2
5. izpis x
```

V prvi vrstici se nastavi spremenljivka  $x$  na 0. V drugi vrstici se najprej ovrednoti desna stran ( $x+2$ , torej  $0+2$ ). Vrednost 2 se priredi spremenljivki  $x$ . V tretji vrstici se spremenljivki  $x$  prišteje 2, torej je  $x$  sedaj enak 4. Podobno velja za četrto vrstico. V peti vrstici se izpiše vrednost 6.

## 5.1 Prečrtaj odvečne vrstice

### 5.1.1 Pravila igre

Igra "Prečrtaj odvečne vrstice" skriva odvečne ukaze - ukaze, ki ne spremenijo rezultata programa. V praksi so takšni ukazi pogosto posledica spreminjanja in dopolnjevanja programov. Za primer vzemimo program, ki je sestavljen iz treh ukazov (vsak ukaz pišemo v svojo vrstico):

#	Program	Izpis
1	<b>x = 3</b>	
2	<b>x = 4</b>	
3	izpis x	4

Če sedaj interpretiramo program lahko rečemo, da v prvi vrstici (**x = 3**), dobi spremenljivka  $x$  vrednost 3. V drugi vrstici (**x = 4**), se spremenljivki  $x$  dodeli nova vrednost, tj. 4. S tem ukazom zamenjamo staro vrednost, ki je sedaj za nadaljnje izvajanje programa "izgubljena". V tretji vrstici ukaz **izpis x**, izpiše vrednost spremenljivke tj. 4.

Izkaže se, da je prva vrstica nepotrebna, saj ne vpliva na izpis in delovanje programa. Zato prečrtamo prvo vrstico:

#	Program	Izpis
1	<del><b>x = 3</b></del>	
2	<b>x = 4</b>	
3	izpis x	4

V drugem zgledu imamo program, ki ima spremenljivki  $x$  in  $y$ .

#	Program	Izpis
1	$x = 7$	
2	$y = 5$	
3	$y = x + 4$	
4	izpis $x$	7

V prvi in drugi vrstici se nastavijo vrednosti spremenljivk  $x$  in  $y$ . V tretji se izračuna nova vrednost za  $y$ , pri tem pa se ne upošteva prejšnja vrednost, tj. 5. Zato lahko trdimo, da je druga vrstica odveč. V zadnji vrstici izpišemo vrednost spremenljivke  $x$ , tj. 7. Ker nam do konca programa spremenljivka  $y$  ni vplivala na izhod programa (izpis v našem primeru), lahko prečrtamo tudi tretjo vrstico. Dobimo:

#	Program	Izpis
1	$x = 7$	
2	<del><math>y = 5</math></del>	
3	<del><math>y = x + 4</math></del>	
4	izpis $x$	7

Za tretji zgled vzemimo malce spremenjen program:

#	Program	Izpis
1	$x = 7$	
2	$y = 5$	
3	$x = x - y$	
4	izpis $x$	2

V prvi in drugi vrstici se nastavijo vrednosti spremenljivk  $x$  in  $y$ . V tretji, spremenjeni vrstici, se izračuna nova vrednost za  $x$ , pri tem pa se upošteva vrednost spremenljivk  $x$  in  $y$  iz vrstic ena in dva. V zadnji vrstici se izpiše vrednost spremenljivke  $x$ , tj. 2. Program torej nima odvečnih ukazov.

**Namig:** Ob ukazih si zapišite tabelo trenutnih vrednosti spremenljivk.

## 5.1.2 Preizkusi se

Če obstajajo odvečne vrstice jih prečrtaj, in v sive celice zapišite izpis programa.

#	Program	Izpis
a) 1	x = 7	
2	y = 9	
3	izpis y	

★

#	Program	Izpis
b) 1	x = 7 + 3	
2	y = x	
3	izpis y	

★★

#	Program	Izpis
c) 1	x = 3	
2	y = 2	
3	izpis x	
4	izpis y	
5	x = 13	

★★

#	Program	Izpis
č) 1	x = 3	
2	y = 4	
3	z = x	
4	x = y	
5	y = z	
6	izpis x	
7	izpis y	

★★★

## 5.2 Vstavi spremenljivko

V prejšnji igri smo v sive celice vpisali vrednosti, ki jih program izpiše. Sedaj bodo te vrednosti že podane, v celice pa je potrebno vpisat, imena spremenljivk tako, da bo program izpisal že podano vrednost (črna celica). Za primer vzemimo program:


#	Program	Izpis
1	<code>x = 2</code>	
2	<code>y = 5</code>	
3	<code><span style="background-color: #cccccc;">?</span> = x + y</code>	
4	<code>izpis y</code>	<span style="background-color: #000000; color: #ffffff;">7</span>

V tretji vrstici, na mesto vprašaja, lahko vpišemo spremenljivko  $x$  ali  $y$  (izbiramo samo med spremenljivkami, ki se pojavijo v programu). V primeru, da v sivo celico vpišemo spremenljivko  $x$ , bo program izpisal vrednost 5 ( $y = 5$ ), kar je narobe, ker želimo da izpiše 7 (glej vrstico štiri). V primeru, da vpišemo spremenljivko  $y$ , pa program pravilno izpiše 7. Pravilno je torej:


#	Program	Izpis
1	<code>x = 2</code>	
2	<code>y = 5</code>	
3	<code><span style="background-color: #cccccc;">y</span> = x + y</code>	
4	<code>izpis y</code>	<span style="background-color: #000000; color: #ffffff;">7</span>

Vstavi pravilne spremenljivke.



★

#	Program	Izpis
1	x = 2	
2	y = 5	
3	 = x + y	
4	izpis x	7

★★

#	Program	Izpis
1	x = 2	
2	y = x + 1	
3	 = y - x	
4	izpis x	1
5	izpis y	3

★★★

#	Program	Izpis
1	x = 2	
2	y = 5	
3	 = x + y	
4	 = x - y	
4	x = x - y	
5	izpis x	5
6	izpis y	2

Vstavi pravilne spremenljivke.

a)

#	Program	Izpis
1	x = 2	
2	y = 5	
3	z = x	
4	<div></div> = y	
5	<div></div> = z	
6	izpis x	5
7	izpis y	2

b)

#	Program	Izpis
1	x = 2	
2	y = x	
2	z = y + 1	
3	<div></div> = y * y	
4	izpis x	2
5	izpis y	4
5	izpis z	3

c)

#	Program	Izpis
1	x = 1	
2	y = 2	
3	z = y	
4	<div></div> = x	
5	<div></div> = z	
6	izpis x	2
7	izpis y	1



Več nalog različnih težavnostnih stopenj lahko najdete v prilogi na straneh od 150 do 154.

## 5.3 Primeri iz prakse

Poglejmo si primer uporabe spremenljivk in vnosa v različnih programskih jezikih. Definirali bomo celoštevilčno spremenljivko z imenom **x** in jo prebrali iz tipkovnice, v našem zapisu uporabimo ukaz:

**vnos x.**

Programski jezik	Ukaz
Pascal	<b>var x:integer;</b> //Definiranje spremenljivke <b>readln(x);</b> //Branje niza
Java	<b>int x;</b> //Definiranje spremenljivke <b>x = Integer.parseInt(in.readLine());</b>
C#	<b>int x;</b> //Definiranje spremenljivke <b>x = System.Convert.ToInt32(Console.ReadLine(),5);</b>
C++	<b>int x;</b> //Definiranje spremenljivke <b>cin &gt;&gt; x;</b> //Branje spremenljivke
Ruby	//Definiranje spremenljivke ni potrebno <b>s = gets.chomp.to_i</b> //Branje niza

V tabeli vidimo, da v različnih programskih jezikih, za isti izpis uporabljamo različne ukaze, vendar vsi delujejo podobno. Pri večini programskih jezikov velja, da moramo spremenljivki določiti tip in ime. Tip določa ali gre za celo število, realno število, niz itd. Za ime pa velja, da je sestavljen iz znakov in števil, pri tem pa ime ne sme vsebovati posebnih znakov, kot so presledek, plus, zvezdica, itd.

*progo.crepinsek@gmail.com*

---

# 6.

*Če se jih veliko vpleta, se posel zaplete.*

*- Latinski pregovor -*

## Vejitve

### Znano

- ★ Točki, kjer se veja deli na dve manjši, pravimo vejitev.
- ★ Križišče oblike Y, je primer vejitve ceste.
- ★ Delitev knjige na poglavja je primer vejitve.
- ★ Človeško telo je prepleteno z mnogimi vejitvami, kot so: živci, žile, delitev sapnika itd.

Vejitev je močno povezana z našo vizualno predstavo. Vejitev zapisana v obliki teksta nam mnogokrat predstavlja težave pri sledenju in razumevanju vsebine. Kot primer navedimo zakone, ki so prepleteni z najrazličnejšimi vejitvami, tj. sklici na različne člene in odstavke, kar mnogokrat privede do različnih interpretacij zakona. Potreba po jasnejšem podajanju vejitev je privedla do različnih grafičnih ponazoritev. Na področju računalništva se je med prvimi uveljavila predstavitev s pomočjo diagramov poteka, ki se danes uporablja tudi na drugih področjih.

## 6.1 Diagram poteka

Diagrame poteka sta prvič predstavila Herman Goldstine in John von Neumann na univerzi Princeton University, leta 1947.

Diagram potek je graf, sestavljen iz določenih grafičnih simbolov: ovalna oblika, trapez, pravokotnik, romb (kara) in puščica.

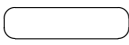
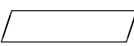
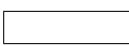


Ime simbola	Namen	Slika
začetni, končni simbol	označuje začetek in konec diagrama	
vhodno izhodni simbol	vnos, izpis	
procesni simbol	vsebuje akcije, izraze	
odločitveni simbol	vsebuje pogoj in opisuje vejitve	
povezava	povezuje simbole	

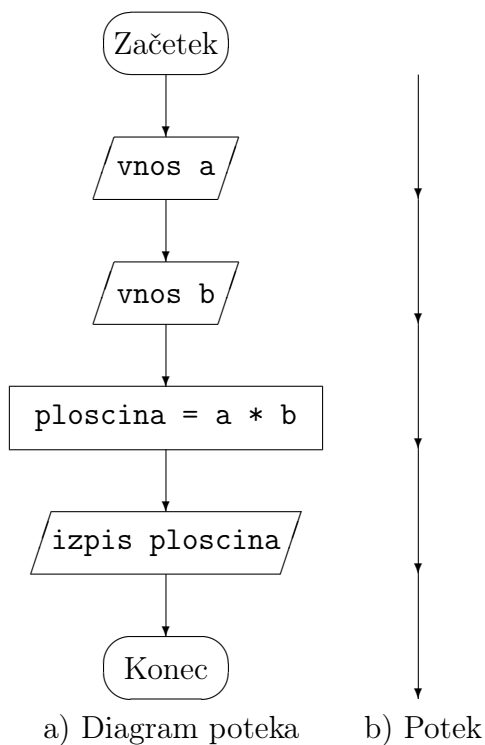
Tabela 6.1: Simboli diagrama poteka

Posamezne simbole diagrama poteka in njihov namen opisuje tabela 6.1. Pri povezovanju grafičnih simbolov, moramo upoštevati naslednja pravila:

- Vhodni, izhodni in procesni simboli imajo lahko samo en izhod (puščica, ki kaže ven).
- Iz odločitvenega simbola potekajo vedno dve povezavi oz. puščici (glede na pogoj), ki so označene z: *potem* in *drugače* oz. *DA* in *NE*.

- V začetni simbol ne pelje nobena puščica.
- Iz končnega simbola ne pelje nobena puščica.
- Vsaka povezava ima smer poteka v diagramu, prikazana s puščico.

Natančnejšo uporabo vseh simbolov bomo spoznali postopoma tekom naslednjih poglavij.

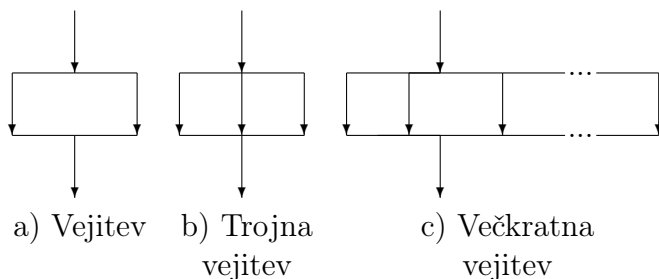


Slika 6.1: Linearni potek programa

V dosedanjih poglavjih smo imeli opravka s seznamom ukazov, ki so se izvršili od prvega do zadnjega. Primer diagrama poteka za program iz prejšnjega poglavja prikazuje slika 6.1 a). Slika b) prikazuje samo tek programa, kjer je vidno da naš program ni imel nobenih vejitev.

Potek takšnega programa, bi lahko primerjali z vožnjo po enosmerni cesti. Ob zagonu, začnemo vožnjo na začetku ceste, ob koncu ceste pa se program ustavi. Vožnja po vedno eni in isti cesti bi močno omejila uporabo takšnih cest. Zato uporabljamo križišča, ki jih v primeru poteka programa imenujemo vejitve.

V primeru križišča se moramo odločiti za eno izmed poti. Pri programu pa to pomeni, da se izvršijo različni bloki (deli) programa.



Slika 6.2: Različni načini vejitev

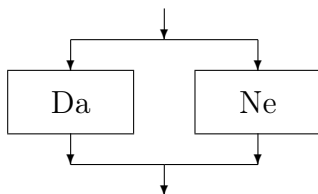
Potek programa lahko vejimo na več načinov: z dvojno, trojno ali večkratno vejitvijo (slika 6.2). V nadaljevanju si pogledjmo nekaj primerov vejitev.

## 6.2 Primeri

Na začetku poglavja bomo obravnavali samo vejitve brez pogojev (poenostavljene diagrame poteka).

### 6.2.1 Tipske vejitve

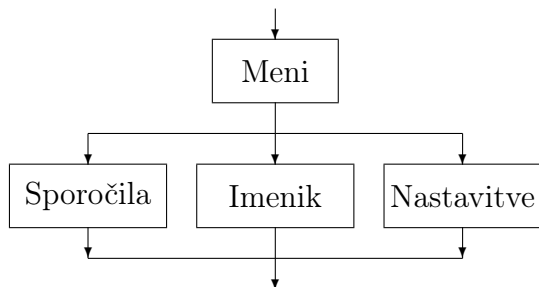
Eno izmed najpogostejših primerov vejitev so odgovori oblike DA ali NE (slika 6.3). Pri vejitvi se moramo odločiti za eno



Slika 6.3: Vejitev Da/Ne

izmed poti, torej DA ali NE.

Znan primer večkratne vejitve je meni mobilnega telefona, kjer izbiramo med različnimi opcijami (slika 6.4)

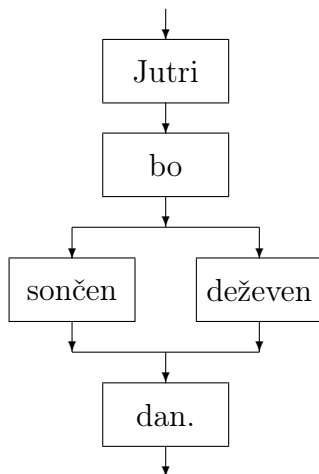


Slika 6.4: Meni na mobilnem telefonu

Pri menijih izbiramo eno izmed operacij, ki so podane v obliki besedila ali slike. Pogosto se za izbrano operacijo

skriva podrobnejši meni oz. nova vejitev.

V pogovoru pogosto uporabljamo tipske stavke, torej stavke ki se razlikujejo samo po manjšem delu (slika 6.5). Pomen takšnih stavkov je lahko popolnoma različen.



Slika 6.5: Napoved vremena

Pomembno je, da lahko ob enem zagonu izberemo samo eno pot od začetka do konca. V primeru iz slike 6.5 lahko napovemo lep ali deževen dan, torej stavek "Jutri bo sončen deževen dan." ni mogoč.

## 6.3 Zapiši možne poti

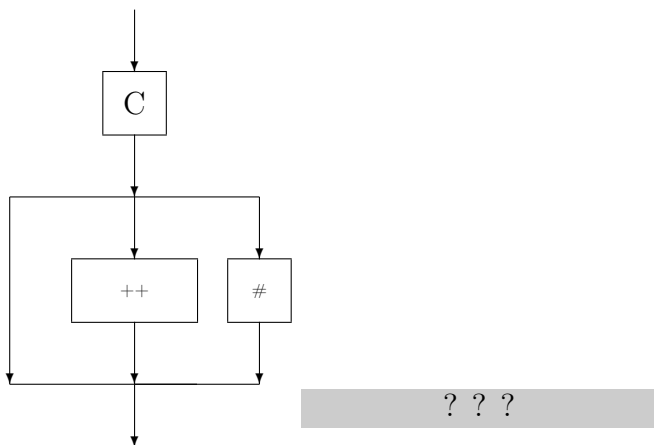
Cilj igre "Zapiši možne poti" je, da se zapišete vse možne poti izvajanja programa.



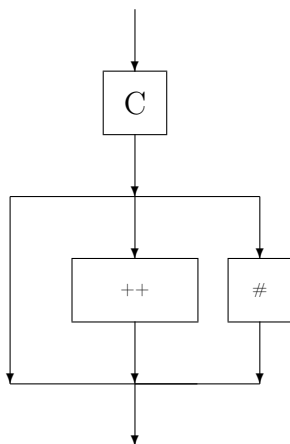
### 6.3.1 Sestavljanje besed

Vejitev se lahko uporablja pri preverjanju besed, tj. prepoznavanju pravilnosti besedila.

Za primer vzemimo spodnji diagram poteka, kjer so zapisane izpeljanke programskega jezika C. Cilj igre je, da v sivo celico zapišite vse besede, ki jih je možno sestaviti s pomočjo diagrama.



Rezultat zgornje igre so imena programskih jezikov: *C*, *C++* in *C#*. Zapis jezika se vedno začne z znakom "C" (zgornji kvadrat), nato lahko sledi: "nič", ++ ali # (glej diagram). Pravilen odgovor zapišemo v sivo celico kot:

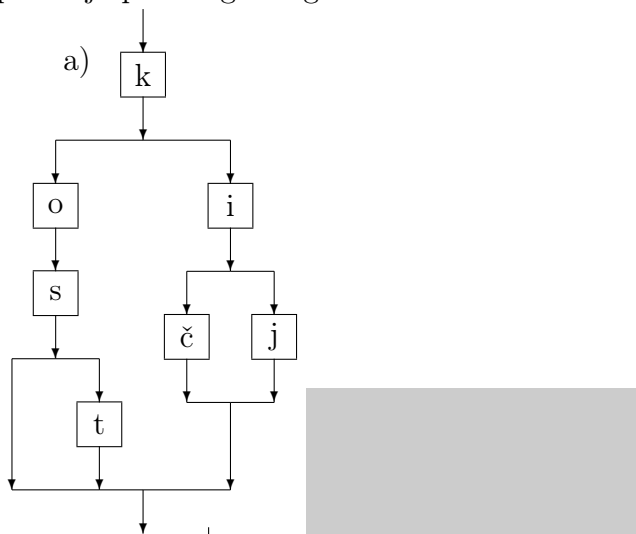


↓

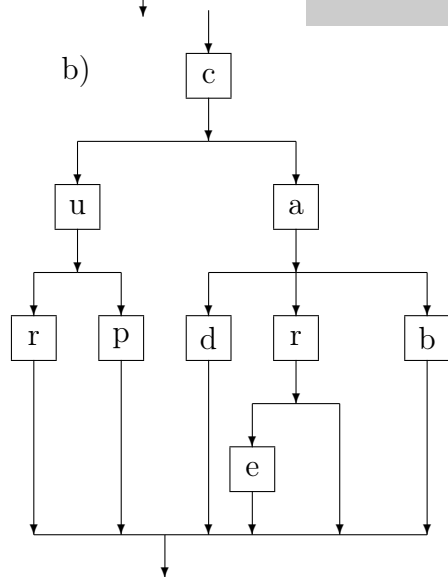
↓

C, C++ ali C#

V sivo celico zapiši vse besede, ki jih je možno sestaviti s pomočjo podanega diagrama.



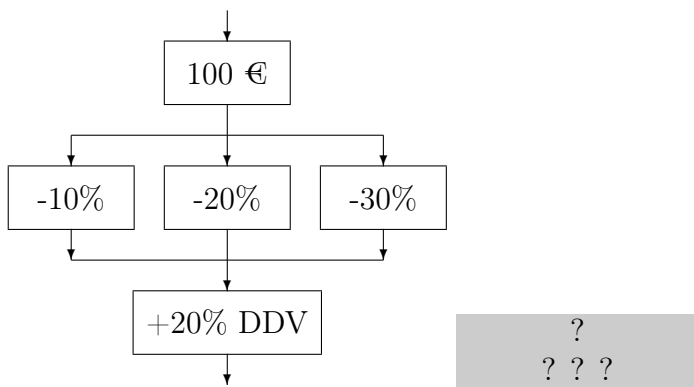
★



★★

### 6.3.2 Izračunaj možne rezultate

S pomočjo vejitev, lahko opišemo različne postopke izračunov. Za primer vzemimo nalogo, katere cilj je, da v sivo polje vpišemo vse možne končne cene torbe, ki ima izhodiščno ceno 100 €. Kupec dobi 10, 20 ali 30% popusta na osnovno ceno. Na koncu moramo dodati še 20% davek. Ocenitev opišemo s pomočjo spodnjega diagrama.



Če izračunamo vse možne cene torbe, dobimo:

- $100 - 10\% + 20\% = 108$  €,
- $100 - 20\% + 20\% = 96$  € ali
- $100 - 30\% + 20\% = 84$  €.

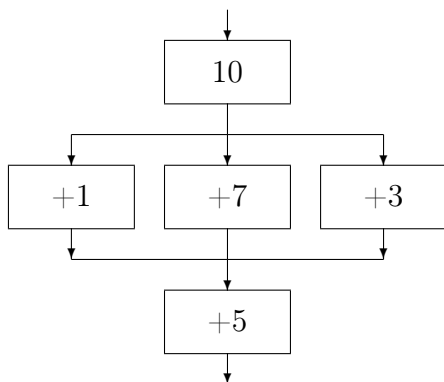
Pravilni odgovor je: 108, 96 ali 84

Prikazovanje postopkov različnih izračunov, s pomočjo diagrama poteka, je mnogo bolj razumljivo kot samo besedilo. Iz diagrama je takoj razvidno, da vsak kupec dobi vsaj 10% popusta.

V sivo celico vpišite vse možne rezultate diagrama poteka. Prvi kvadrateg v diagramu opisuje začetno vrednost.

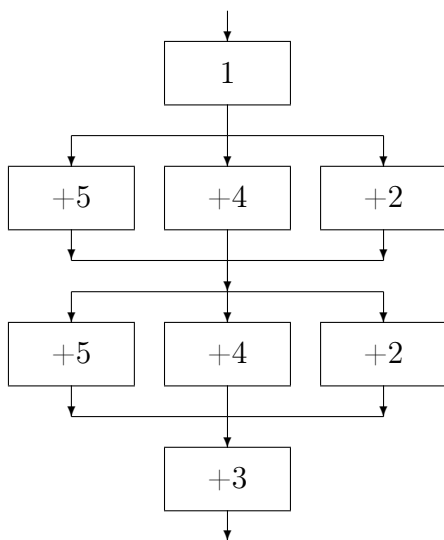
a)

★★★



b)

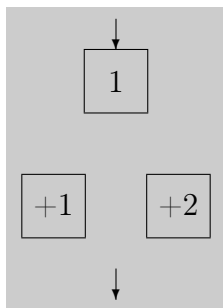
★★★



## 6.4 Poveži diagram

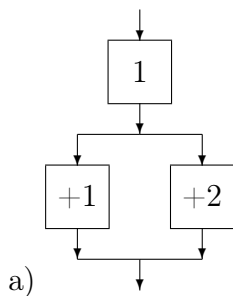
Poveži diagram poteka tako, da bodo možni samo podani rezultati (črna celica desno). Prvi kvadrateg v diagramu opisuje začetno vrednost. Sledi primer z rešitvijo.

Pravilno poveži diagram:



2, 3

Pravilen odgovor je:

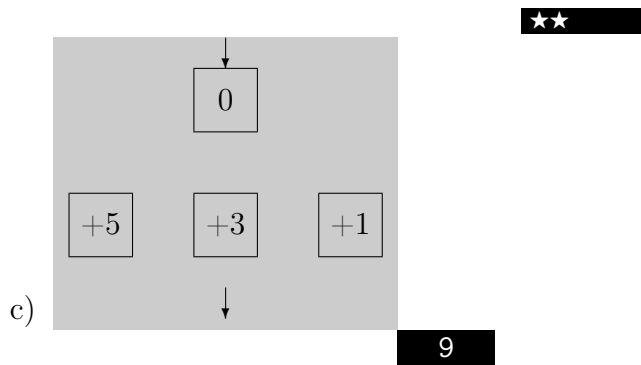
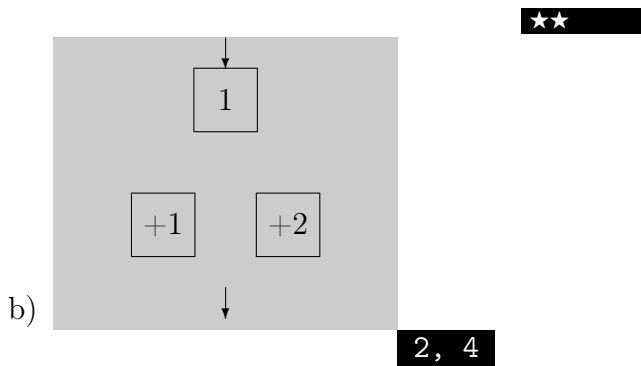
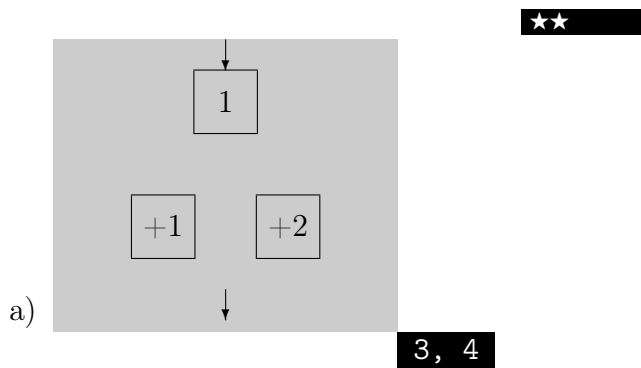


2, 3

**Opomba:** Pri povezovanju moramo biti pozorni na smer puščic in katere kvadratke povezujemo.

**Namig:** Število različnih možnih rezultatov (črna celica), opisuje koliko različnih simbolov diagrama je povezanih s koncem diagrama.

Poveži diagram tako, da dopušča samo določene rezultate.



### 6.4.1 Izpolnjevanje obrazcev

Izpolnjevanje obrazcev je v osnovi preprost, v praksi pa pogosto zapleten postopek. Težave nastopijo zaradi nepoznavanja vsebine, ki jo narekuje posamezno polje.

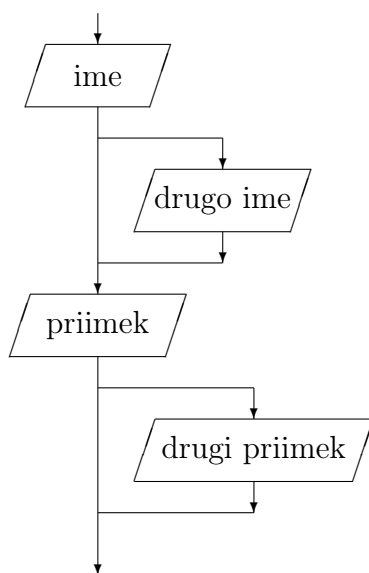
Za primer vzemimo najpogostejša polja, ki jih srečamo na obrazcih, tj. polja "ime" in "priimek". V osnovi pri teh poljih ni težav, a tudi ta imajo svoje posebnosti. Vsak državljan Evropske skupnosti, ima lahko do dva imena in do dva priimka. Takšen obrazec ima lahko naslednjo obliko:

Prvo ime:	_____
Drugo ime:	_____
Prvi priimek:	_____
Drugi priimek:	_____

Pri takšnem obrazcu je mnogo ljudi za trenutek negotovih, saj nimajo drugega imena in drugega priimka. Naravna želja izpolnjevalca pa je, da bi izpolnil vsa polja. S pomočjo diagrama poteka vnosa lahko hitro opazimo, da sta drugi ime in priimek opsijska (slika 6.6).

Vejitve, ki smo jih do sedaj opisovali v diagramih poteka, opisujejo možne poti. Ne opisujejo pa kdaj se odločimo za katero izmed njih. Kako opisati izbiro posamezne poti, opisuje naslednje poglavje.





Slika 6.6: Vnos imena in priimka

## 6.5 Odločanje

*Izbira je zadrega. - Nemški pregovor -*

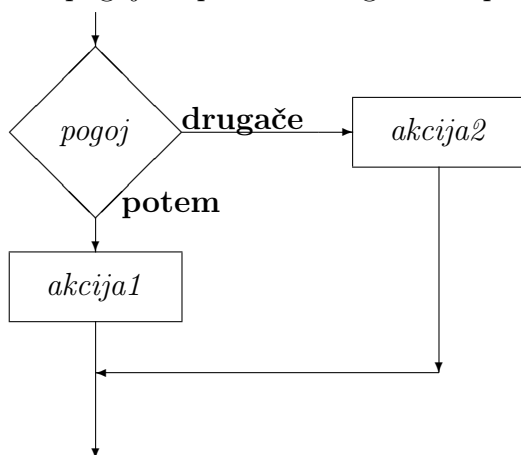
Vsaka vejitev nam omogoča izbiro poti ali izvršitev dela programa. Za "pot" se možgani odločajo na podlagi izkušenj in cilja, v nasprotnem primeru izberemo naključno "pot". Da se program ne bi odločal naključno, mu moramo pred vejitvijo opisati pogoje, pod katerimi se bo odločil za eno izmed "poti". Kot primere odločanja naštejmo nekaj pogojev iz vsakdanjega življenja.

- Če bo lepo vreme, gremo v gozd, sicer v kino.
- Če bomo dovolj hitri, gremo na vlak danes, drugače bomo prespali v mestu.
- Če bo dovolj denarja, bom kupil nov avto.
- Če bom zbral vse nalepke, bom dobil nagrado.
- Če bom razumel to poglavje, grem na naslednjega.
- :

Iz podanih pogojev lahko zapišemo splošno obliko zapisa takšnih in podobnih pogojev:

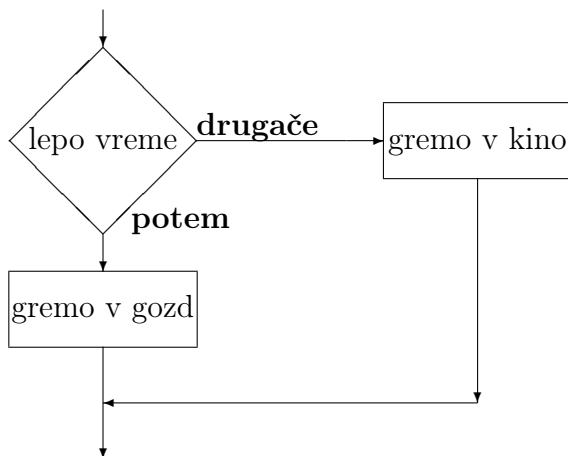
**Če pogoj potem akcija1 drugače akcija2.**

Splošna obliko pogoja zapišimo z diagramom poteka:



Slika 6.7: Splošen opis pogoja

Na diagramu vidimo, da smo spustili besedico "če", ki jo nadomesti pogojni simbol v obliki kare (romba). Zapišimo pomen stavka, "*Če bo lepo vreme, gremo v gozd, drugače gremo v kino.*", s pomočjo diagrama.



Slika 6.8: Opis pogojnega stavka

Pri pogoju imamo vedno možnosti: **potem** in **drugače**. Zaradi krajšega zapisa bomo v diagramu pisali namesto **potem**, **da**; in namesto **drugače** **ne**. Podobno imamo dve možnosti pri logičnih izrazih, kjer je izračunana vrednost vedno samo **resnično** ali **neresnično**. Pri logičnih izrazih se uporabljajo operatorji kot so: ali, in, negacija, različno, enako, večje, manjše, itd. Rezultat logičnega izraza je lahko samo resnično in neresnično.

Ker se računalnik ne zna odločati pri pogojnih stavkih, kot so: Je lep dan?, Je večji?, Je dovolj velik?, Imamo dovolj denarja? itd., uporabimo namesto pogoja logične izraze. Kako lahko zapišemo logičen izraz, namesto opisnega pogoja vidimo na naslednjih primerih.

Tabela 6.2: Splošen pogoj s pripadajočim logičnim izrazom

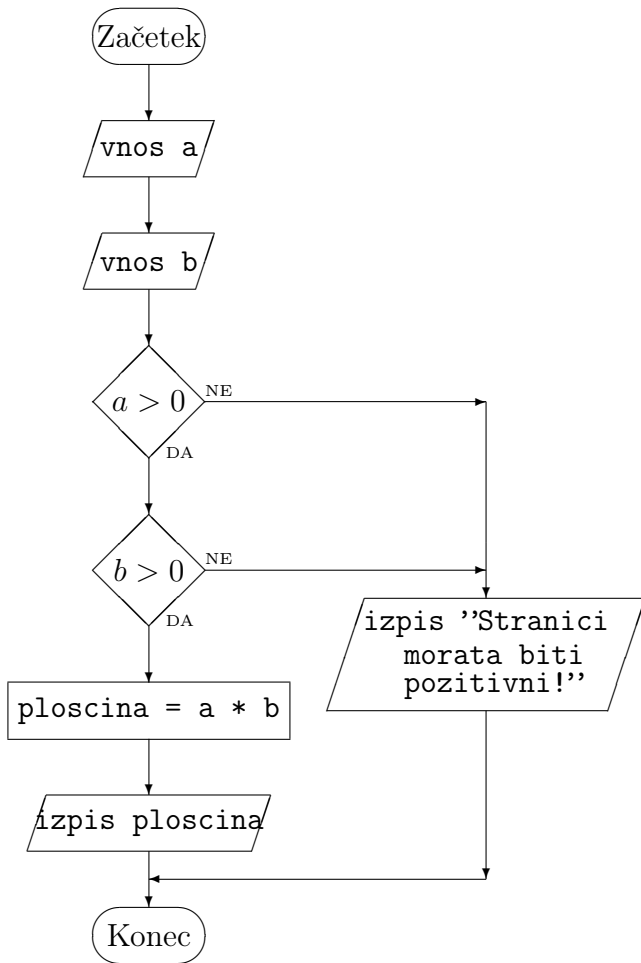
#	Opisni pogoj	Logični izraz
1	večji od 120cm	<code>visina &gt; 120</code>
2	večji od Martina	<code>visina &gt; visinaMartin</code>
3	dovolj denarja	<code>cena &lt;= razpoložljivDenar</code>
4	v petek	<code>dan == "Petek"</code>
5	lepo vreme	<code>vreme == "Sončno"</code>
6	če je 7+11 večje od 17	<code>(7+11) &gt; 17</code>
7	cena različna od 100	<code>cena &lt;&gt; 100</code>

V tabeli na strani logičnih izrazov uporabljamo spremenljivke in konstante, katerih vrednost mora biti v času odločanja znana. V prvem primeru uporabnik vnese podatek o višini, ki mora biti v centimetrih. Pri drugem primeru uporabnik vnese dve višini, Martinovo in višino osebe, ki jo hoče primerjati z Martinovo. Pri tretjem primeru moramo vedeti koliko denarja imamo na razpolago (`razpoložljivDenar`) in ceno tistega kar želimo. Pri četrtem primeru moramo vnesti dan v tednu, ki ga nato primerjamo s konstanto `"Petek"`.

Podobno velja pri petem primeru, v šestem pa računalnik v izrazu izračuna vrednost, nato pa primerja vrednosti. V tem primeru ne rabimo nobenih dodatnih spremenljivk, potrebno pa je poudariti, da se bo program vedno odločil enako.

Pisanje logičnih izrazov zahteva zapis, ki je mnogo natančnejši od pogovornega jezika, žal pa je pogosto manj razumljiv.

Na začetku poglavja imamo diagram poteka za izračun ploščine pravokotnika (slika 6.1). Program ima pomanjkljivost, saj lahko za stranici pravokotnika **a** in **b**, vnesemo negativne vrednosti, to pa je fizikalno nemogoče.

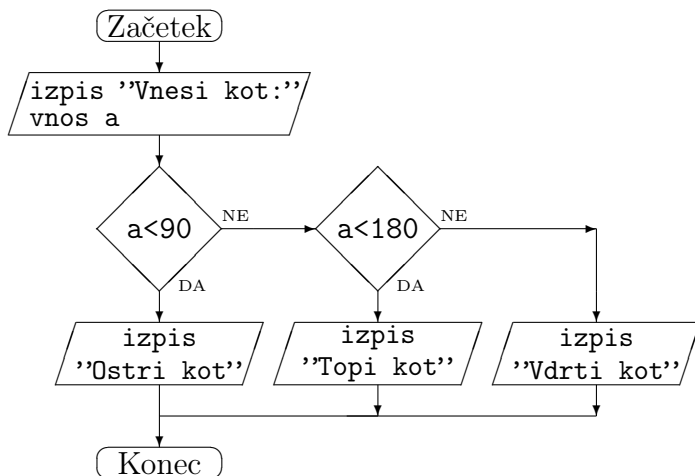


Slika 6.9: Diagram izračuna ploščine pravokotnika

V želji po boljšem programu, popravimo diagram tako, da ob negativnih vrednostih uporabniku izpišemo opozorilo "Stranici morajo biti pozitivni!" (slika 6.9).

## 6.6 Zapiši izpis

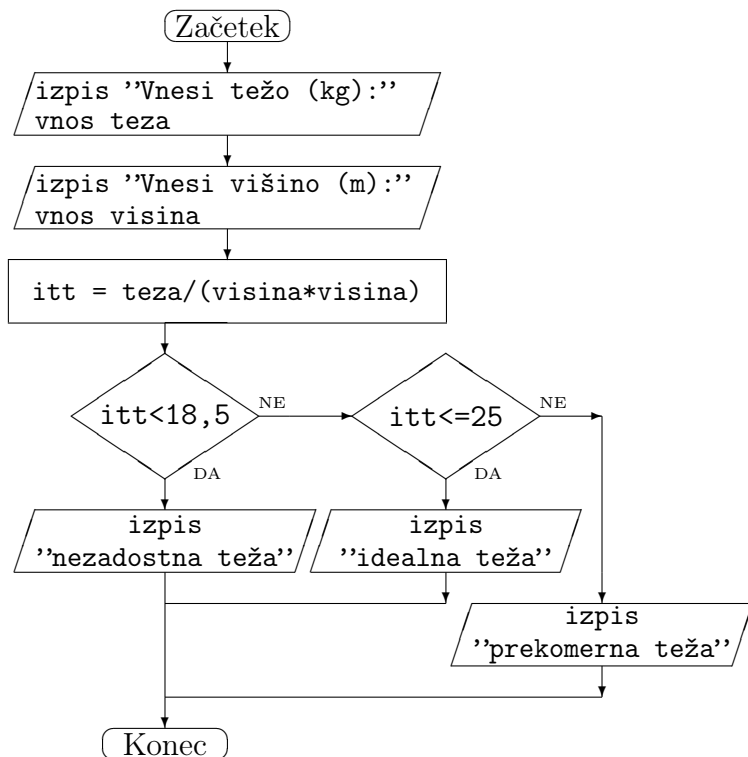
Cilj igre je, glede na podan vnos, določiti izpis programa (glej spodnjo tabelo).



Vhod je podan v obliki tabele (stolpec vnos). Zapisati je potrebno izpis (stolpec izpis). Za primer vzemimo za vhod vrednost 44 (vnos uporabnika), če sedaj sledimo diagramu poteka, vidimo da se izpiše "Ostri kot" (44 je manjše od 90). V primeru vhodne vrednosti 222, se izpiše "Vdrti kot". Dopolni tabelo za ostale vrednosti.

Vnos (a)	Izpis	
44	Ostri kot	
222	Vdrti kot	
a) 2		★★
b) 350		★★
c) 89		★★
č) 91		★★
d) 90		★★★★

Program za izračun indeksa telesne teže (itt).



Dopolnite tabelo s pripadajočim izpisom.

	Vznesena teža	Vznesena višina	Izpis
--	---------------	-----------------	-------

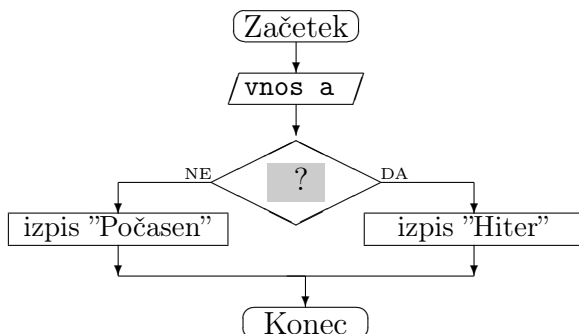
	Vznesena teža	Vznesena višina	Izpis
a)	100	2,00	
b)	60	1,70	
c)	90	1,82	
č)	50	1,67	
d)			

\* Pri nalogi d, vpišite svoje podatke o teži in višini.



## 6.7 Izberi pogoj

Cilj igre je izbrati pogoj-e v diagramu (sive celice v diagramu), tako da dobimo ob podanem vnosu (stolpec vnos) željen rezultat (stolpec izpis). Izbiramo med že vnaprej pripravljenimi pogoji. Kot primer si pogledjmo iskanje pogoja pri vnosu vrednosti 120 ( $a=120$ ) in izpisu "Hiter". Izbiramo med pogoji:  $a>100$ ,  $a>110$ ,  $a<90$  in  $a>130$ .



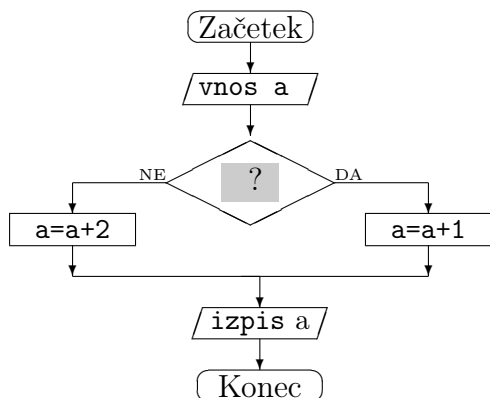
**Namig:** Za vsak pogoj preveri ali drži, da pri podanem vnosu dobimo določen izpis. Če drži, pogoj vpišemo v sivo celico.

Pravilen odgovor je:  $a>100$  ali  $a>110$  (spodaj že izpolnjen).

	Vnos a	Izpis a	Pogoj	
	120	Hiter	$a>100$ ali $a>110$	
a)	101	Hiter		★★
b)	129	Počasen		★★
c)	70	Hiter		★★
č)	150	Počasen		★★

Za drugi zgled pogledjmo diagram, kjer se vrednost spremenljivke  $a$  tekom izvajanja spreminja. Za primer si pogledjmo iskanje pogoja diagrama (siva celica) pri vnosu vrednosti 5 ( $a=5$ ) in izpisu 6.

Izberamo med pogoji:  $a>3$ ,  $a>4$ ,  $a>5$ ,  $a>6$  in  $a>7$ .



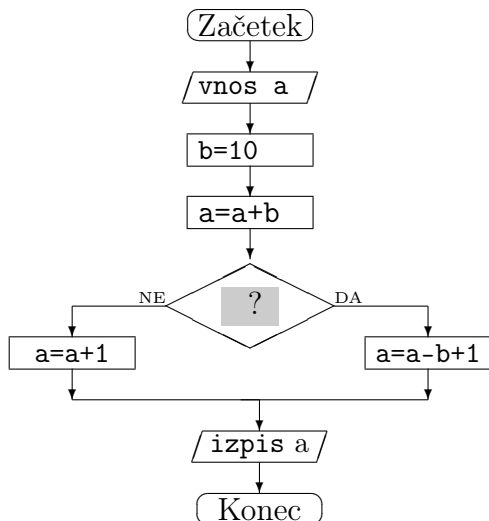
**Namig:** Za lažje reševanje si s svinčnikom ob simbolih diagrama, kjer pride do spremembe vrednosti spremenljivk, ( $a = a + 1$  in  $a = a + 2$ ) zapišemo vmesne rezultate.

V zgornjem primeru, si pri vhodu 5, zapišemo ob simbolu diagrama  $a=a+1$  opombo  $a=6$  in ob simbolu diagrama  $a=a+2$  opombo  $a=7$ .

Pravilen odgovor je  $a>3$  ali  $a>4$  (spodaj že izpolnjen).

	Vnos a	Izpis a	Pogoj	
	5	6	$a>3$ ali $a>4$	
a)	5	7		★★★
b)	6	7		★★★
c)	1	3		★★★
č)	1	2		★★★★

Pravilno izberi pogoj-e, ki te prevede-jo do podanega izpisa.



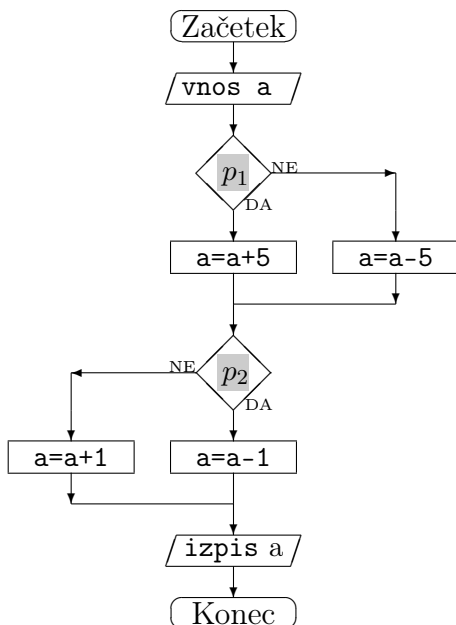
Izberi med pogoji:  $a > 10$ ,  $a > b$ ,  $a < b$  in  $a = b$ .

Pravilen odgovor-e vpiši v sive celice.

	Vnos a	Izpis a	Pogoj	
a)	1	12		★★★
b)	0	1		★★★
c)	1	2		★★★
č)	2	3		★★★
d)	10	11		★★★
e)	10	21		★★★

Ne pozabite izračunati vmesne rezultate (npr.:  $a = a + b$ ) in označite pot po kateri se mora program izvajati, da dobimo željen rezultat. Šele nato preverjajte kateri pogoji ustrezajo izbani poti.

Spodnji diagram ima dve vejitvi, ki sta označeni s  $p_1$  in  $p_2$ . Izberite pare pogojev diagrama  $(p_1, p_2)$  tako, da dobimo ob podanem vходу željen izpis.



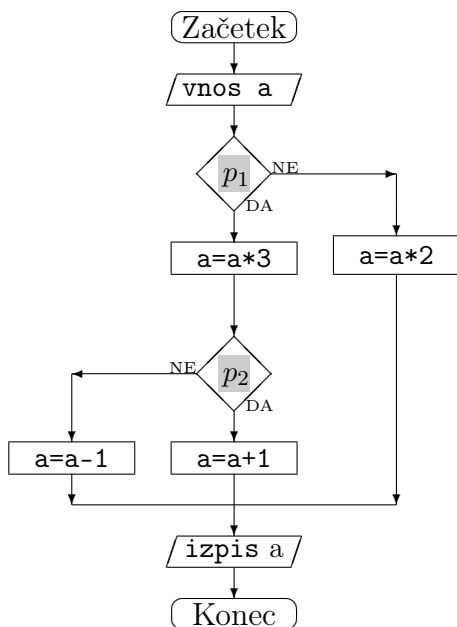
Izberi med pogoji:  $a < 4$ ,  $a > 7$  in  $a == 8$ .

Pravilen odgovor za vход 3 in izpis 7 je izbira za prvo vejitev ( $p_1$ )  $a < 4$  in za drugo vejitev ( $p_2$ )  $a > 7$ , tj. par  $(a < 4, a > 7)$ . Prav tako je pravilen par  $(a < 4, a == 8)$ .

Vnos a Izpis Pogoj  $(p_1, p_2)$

	3	7	$(a < 4, a > 7)$ ali $(a < 4, a == 8)$	
a)	8	14		★★★★★
b)	-5	-1		★★★★★
c)	1	-3		★★★★★
č)	2	6		★★★★★

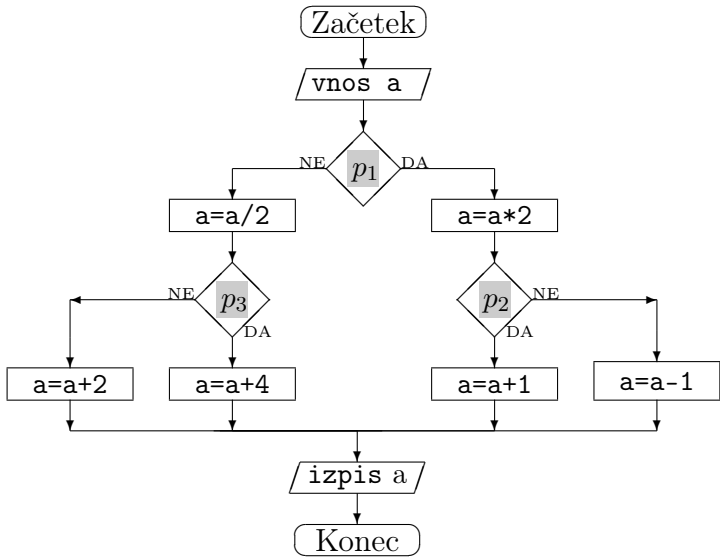
Izberi pare pogojev diagrama  $(p_1, p_2)$  tako, da dobimo ob podanem vnhodu željen izpis.



Izberi med pogoji:

- $a > 3$ ,
- $a == 2$  in
- $a == 12$ .

	Vnos a	Izpis a	Pogoj $(p_1, p_2)$	
a)	2	7		★★★★
b)	4	11		★★★★
c)	4	13		★★★★
č)	12	35		★★★★
d)	12	37		★★★★
f)	1	2		★★★★

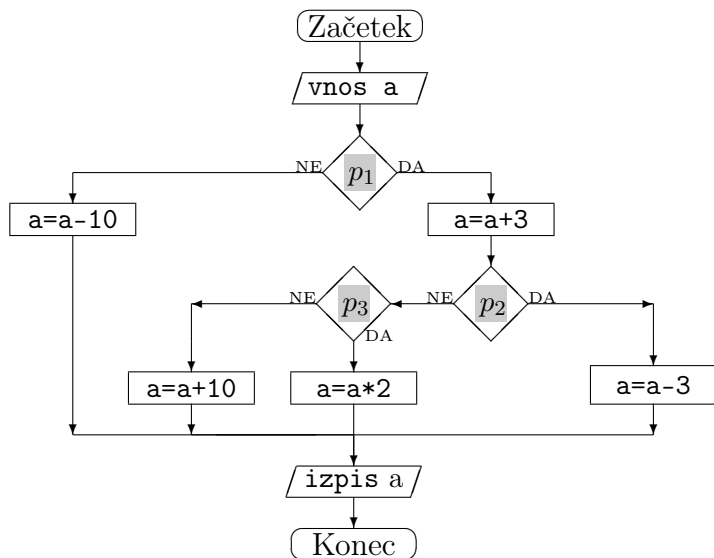


V desni tabeli poveži oznake vejitev diagrama ( $p_1, p_2 \dots p_n$ ) s pripadajočim pogojem (stolpec Pogoj), tako da bo program ob podanem vhodu (stolpec Vnos a), izpisal podan rezultat (stolpec Izpis a). Vsak pogoj lahko uporabite samo enkrat.

Vnos a	Izpis a	Oznaka vejitve	Pogoj
2	3	$p_1$	$a > 1$
4	7	$p_2$	$a > 3$
		$p_3$	$a > 15$
★★★★★			

a)	Vnos a	Izpis a	Oznaka vejitve	Pogoj
	2	3	$p_1$	$a > 1$
	4	6	$p_2$	$a > 3$
			$p_3$	$a > 15$
★★★★★				

b)	Vnos a	Izpis a	Oznaka vejitve	Pogoj
	2	3	$p_1$	$a > 1$
	4	9	$p_2$	$a > 3$
			$p_3$	$a > 15$



Poveži oznake vejitev s pripadajočim pogojem, tako da bo program ob podanem vходу, izpisal željen rezultat. Vsak pogoj lahko uporabite samo enkrat.

★★★★			
	Vnos a	Izpis a	Oznaka vejitve      Pogoj
a)	11	11	$p_1$ $a > 5$
	12	2	$p_2$ $a < 12$
			$p_3$ $a == 14$
★★★★			
	Vnos a	Izpis a	Oznaka vejitve      Pogoj
b)	6	6	$p_1$ $a > 5$
	11	28	$p_2$ $a < 12$
			$p_3$ $a == 14$
★★★★			
	Vnos a	Izpis a	Oznaka vejitve      Pogoj
c)	4	14	$p_1$ $a > 5$
	12	2	$p_2$ $a < 12$
			$p_3$ $a == 14$

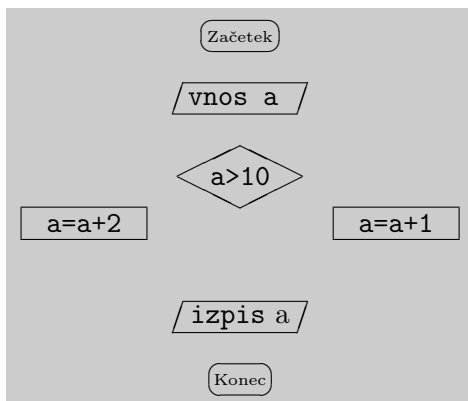
## 6.8 Poveži

S puščicami poveži simbole diagrama tako, da bo program ob podanih vseh vrnil željen rezultat in označite vejitve z DA in NE. Za primer povežimo spodnji diagram tako, da ob vnosu števila 9 izpiše 10 in ob vnosu števila 11 izpiše 13.

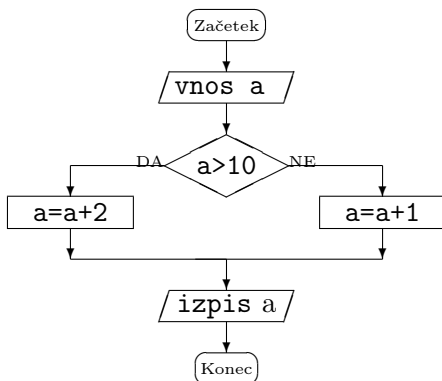
Vnos Izpis

9 10

11 13



Pravilno povezan in označen diagram je:



Zgornji primer je sorazmerno lahek, saj vsebuje samo eno vejitev in logično razporeditev simbolov diagrama. V bodoče velja, da simboli niso vedno logično razporejeni, zato si bomo pomagali z daljšimi potmi puščic.



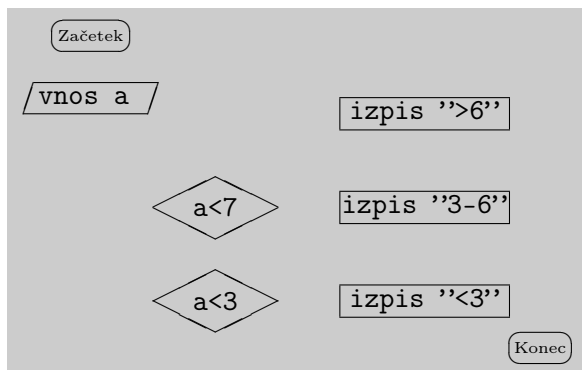
## Pravila povezovanja diagrama poteka

Pri povezovanju ne pozabimo na pravila, ki veljajo za diagram poteka:

- Začetni simbol mora imeti en izhod (puščico iz začetnega simbola).
- Končni simbol mora imeti en vhod (puščico, ki kaže v končni simbol).
- Vsak procesni simbol (kvadrat), mora imeti en izhod.
- Vsak odločitveni simbol (romb), mora imeti dva izhoda, ki sta označena z DA in NE.
- Vsak simbol diagrama, razen začetnega, ki ima izhod mora imeti tudi vhod. V nasprotnem primeru ni del diagrama.

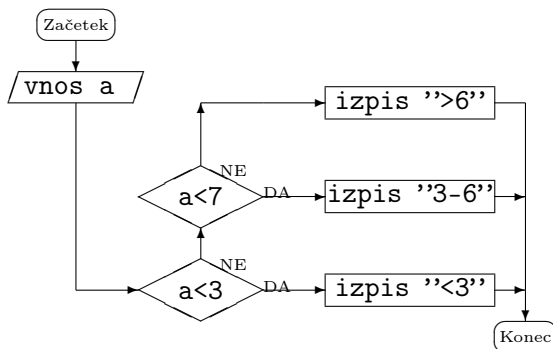
Za boljšo predstavo rešimo še en primer. Povezati moramo dve vejitvi in tri izpise tako, da ustreza pogojem iz podane tabele.

Vnos	Izpis
2	<3
6	3-6
8	>6



Pri povezovanju moramo biti pozorni na potek povezovanja, tj. kateri simboli diagrama so med seboj povezani in v katero smer. Med vašo in podano rešitvijo je lahko vizualna razlika, ki nastane zaradi drugačne izbire strani povezovanja simbolov diagrama.

Pravilno povezan in označen diagram je:



**Namig:** Pred začetkom povezovanje preuči vnos in izpis, ter si poizkusi zamisliti kaj program dela. Nato preuči še podane simbole diagrama in jih poveži.

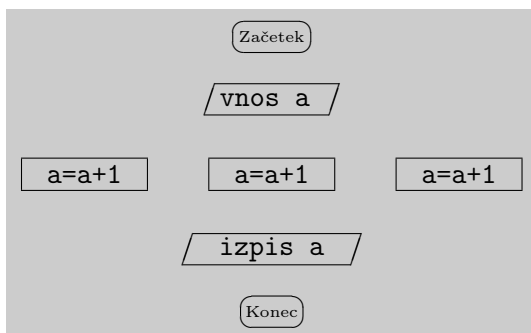
Poveži diagram tako, da ustreza tabeli (ponekod ni potrebno povezati vseh simbolov).

Vnos Izpis

2	5
5	8

★

a)

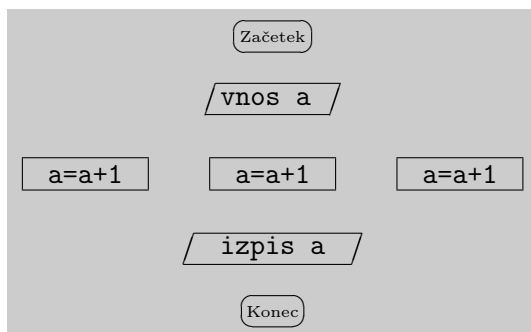


Vnos Izpis

9	11
11	13

★

b)

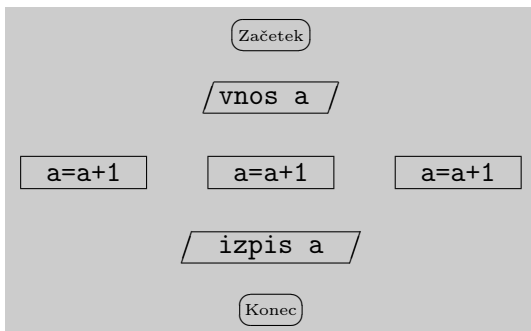


Vnos Izpis

2	2
3	3

★★

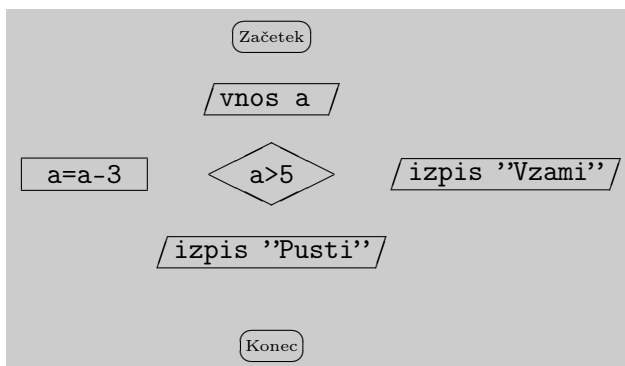
c)



Poveži diagram tako, da ustreza tabeli (ponekod ni potrebno povezati vseh simbolov).

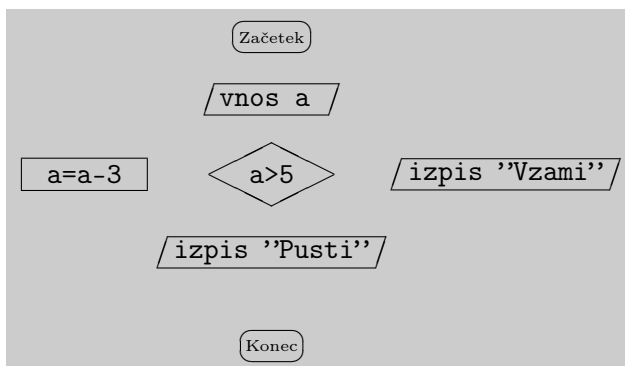
	Vnos	Izpis
a)	9	Pusti
	6	Vzami

★★



	Vnos	Izpis
b)	6	Pusti
	5	Vzami

★★

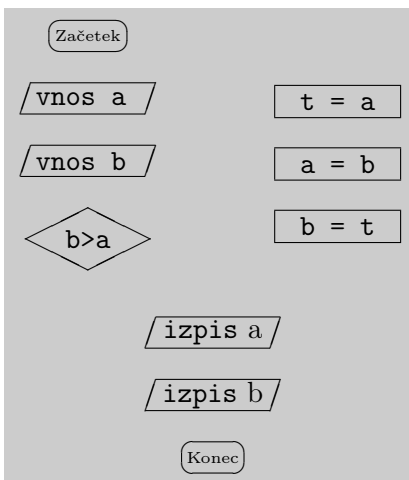


Poveži diagram tako, da ustreza pogojem v tabeli. Program prebere dve spremenljivki (a in b), ter ju na koncu izpiše.

★★★

a)

Vnos		Izpis	
a	b	a	b
2	3	2	3
3	2	2	3
7	1	1	7

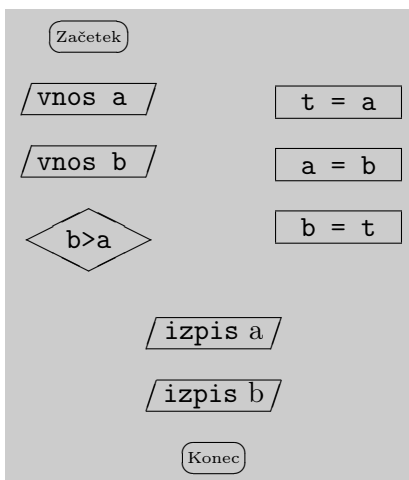


Spodnji primer je podoben prejšnjemu, razlika je pri izpisu.

★★★

b)

Vnos		Izpis	
a	b	a	b
2	3	3	2
3	2	3	2
7	1	7	1

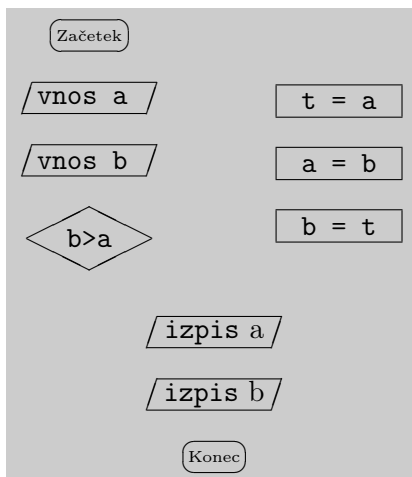


Poveži diagram, tako da ustreza pogojem v tabeli. **Pri povezovanju ni nujno, da uporabite vse simbole diagrama.**

★★★★★

a)

Vnos		Izpis	
a	b	a	b
2	3	2	3
3	2	2	2
7	1	1	1

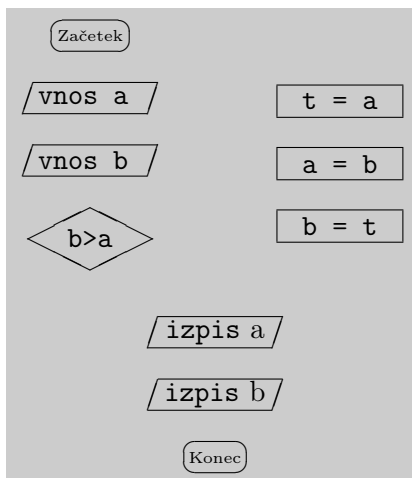


**Namig:** Pri reševanju prvo preučite vhode in izhode, oz. poizkusite ugotoviti kaj naj program dela, šele nato poizkusite pravilno povezat diagram.

★★★★★

b)

Vnos		Izpis	
a	b	a	b
2	3	2	3
3	2	3	3
7	1	7	7

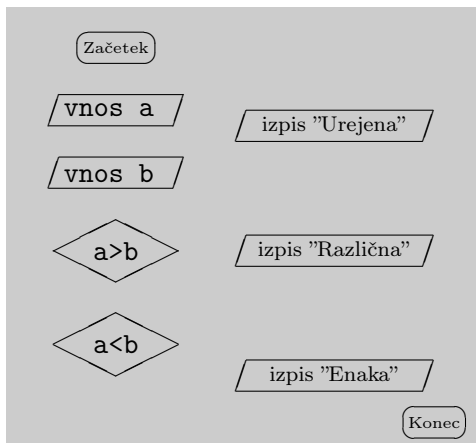


Poveži diagram, tako da ustreza pogojem v tabeli. Pri povezovanju ni nujno, da uporabite vse simbole diagrama.

★★★★

a)

Vnos		Izpis
a	b	
2	3	Različna
3	3	Enaka
7	1	Različna

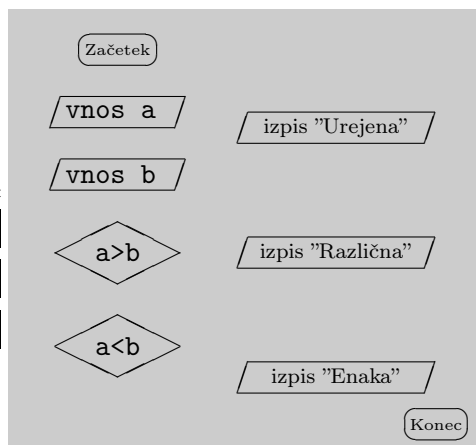


**Namig:** Za preglednejšo rešitev se izogibajte križanju črt.

★★★★

b)

Vnos		Izpis
a	b	
2	3	Urejena
3	3	Enaka
7	1	Različna



Več nalog različnih težavnostnih stopenj lahko najdete v prilogi na straneh od 154 do 176.

## 6.9 Primeri iz prakse

V praksi so vejitve eden izmen najpomembnejših elementov programskih jezikov. Večina programskih jezikov, ki jih danes poznamo, je nastalo na anglosaksonskem področju, zato so za opis posameznih ukazov uporabljali besede iz angleščine. Tako namesto že omenjene splošne oblike:

**Če pogoj potem *akcija1* drugače *akcija2*.**

velja splošna oblika v angleščini (primer za programski jezik Pascal):

**if pogoj then *akcija1* else *akcija2*;**

Pogoj predstavlja logični izraz, ki se s pomočjo procesorja izračuna. Če je izračunana vrednost logičnega izraza enaka nič (neresnično - false), se izvede *akcija2*, drugače *akcija1*. Opazimo, da so snovalci programskih jezikov želeli opisati ukaze v naravnim jeziku podobni obliki. Prav tako je večina programskih jezikov zapisana v tekstovni obliki, zato so nekoliko manj pregledni, v primerjavi z diagramom poteka. Da delno premagamo to pomanjkljivost, si pomagamo s tekstovnim oblikovanjem, kot je delitev ukazov v več vrstic, zamikanje gnezdenja itd. V večini programskih jezikov tekstovna oblika ne vpliva na pravilnost delovanja, to velja tudi za naše vzorčne jezike.

Zaradi preglednosti bomo zapisali pogojni stavek v štirih vrsticah. Pri pogojnih stavkih velja omeniti, da je drugi del ukaza (*else akcija2*) opcijski, torej ga uporabimo po potrebi.



Programski jezik	Ukaz
Pascal	<b>if</b> pogoj <b>then</b> akcija1 //Če drži pogoj se akcija2 preskoči <b>else</b> akcija2; //Če ne drži pogoj se akcija1 preskoči
java	<b>if</b> ( pogoj ) akcija1 ; //Če drži pogoj se akcija2 preskoči <b>else</b> akcija2; //Če ne drži pogoj se akcija1 preskoči
C#	<b>if</b> ( pogoj ) akcija1 ; //Če drži pogoj se akcija2 preskoči <b>else</b> akcija2; //Če ne drži pogoj se akcija1 preskoči
C++	<b>if</b> ( pogoj ) akcija1 ; //Če drži pogoj se akcija2 preskoči <b>else</b> akcija2; //Če ne drži pogoj se akcija1 preskoči
ruby	<b>if</b> pogoj akcija1 //Če drži pogoj se akcija2 preskoči <b>else</b> akcija2 //Če ne drži pogoj se akcija1 preskoči <b>end</b>

Pri tabeli opazimo, da je pogojni stavek v različnih programskih jezikih zelo podoben. Kljub temu pa moramo sintakso poznati zelo natančno, saj je dovolj, da manjka en znak in program nebo deloval.

*progo.crepinsek@gmail.com*

---

*Tudi najbolj spretnemu tkalcu  
se nitka kdaj utrga.*

*- Nemški pregovor -*

# 7.

## Ponavljanje

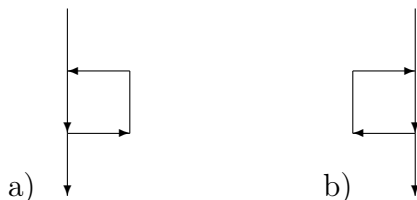
### Znano

- ★ Pri telovadbi večkrat ponovimo določeno vajo.
- ★ Za tekočim trakom se ponavljajo delovne operacije.
- ★ Ponavljanje je sestavni del učenja.

V poglavju "Vejitve" smo primerjali potek programa z vožnjo po enosmerni cesti, vejitev pa je predstavljalo križišče, kjer se na podlagi pogoja odločimo za eno izmed možnih poti. V praksi pogosto naletimo na potrebo, da določen del poti večkrat prepotujemo, kar dosedanji opis ne omogoča, saj zahteva, da vedno odpeljemo celotno pot od začetka do konca.

Ponavljanje lahko ponazorimo na primeru tovornjaka, ki se na začetku delovnega dne odpelje na gradbišče. Na gradbišču prevaža tovor od ene točke do druge, torej celi dan ponavlja del poti. Po končanem delovnem dnevu se tovornjak odpelje domov.

Ponavljanje v diagramu poteka opišemo s pomočjo povratne zanke (slika 7.1).



Slika 7.1: Primera ekvivalentnih povratnih zank

Kadar se odločamo za ponavljanje sta bistveni stvari, kateri del bomo ponavljali in kako dolgo, tj. kdaj bomo prenehali s ponavljanjem. V vsakdanjem življenju pogosto srečamo dva načina podajanja navodil za ponavljanja. V prvem primeru podamo število ponovitev. Npr.:

Naredi 30 počepov.

Odpelji 52 krogov.

...

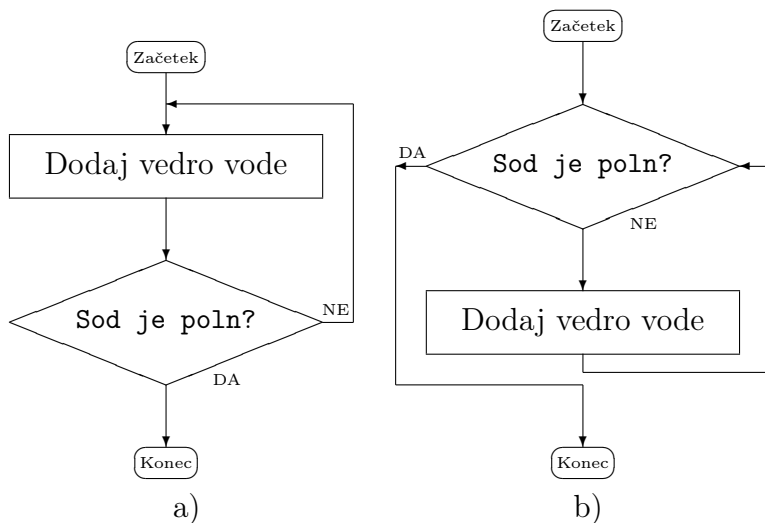
Drugi, prav tako pogost, način opisa ponavljanj je po vzorcu; ponavljaj dokler pogoj ni izpolnjen. Npr.:

Dodajaj vedro vode, dokler ni sod poln.

Vozi zemljo, dokler jama ne bo polna.

...

Ponavljanje lahko opišemo v diagramu poteka s pomočjo vejitve in povratne zanke. Lahko bi rekli, da je zanka posebna vrsta vejitve.



Slika 7.2: Primera opisa ponavljanja

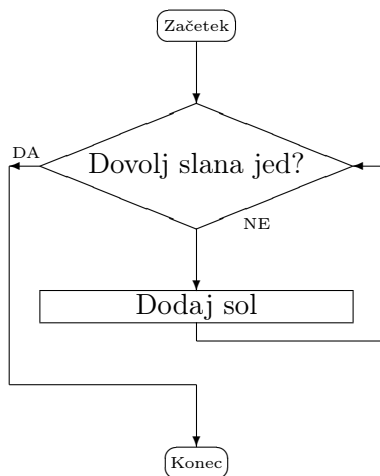
Stavek ki opisuje ponavljanje: "Dodajaj vedro vode, dokler ni sod poln." lahko opišemo s pomočjo diagrama na dva načina (slika 7.2). V prvem diagramu najprej dodamo vedro vode in nato preverjamo ali je poln, v drugem pa prvo preverimo ali je sod poln in šele nato dodamo vodo. V praksi je za dani primer pravilnejši drugi diagram, saj bi sod lahko bil že na začetku poln. Pogosto pa naletimo na primere, kjer si želimo ponoviti del programa vsaj enkrat, takrat uporabimo obliko prvega diagrama.

## 7.1 Primeri

Za boljše razumevanje prikažimo nekaj primerov uporabe ponavljanja.

Kot prvi primer si pogledjmo postopek soljenja jedi. Najprej poizkusimo jed in če ni dovolj slana jo dosolimo. Postopek

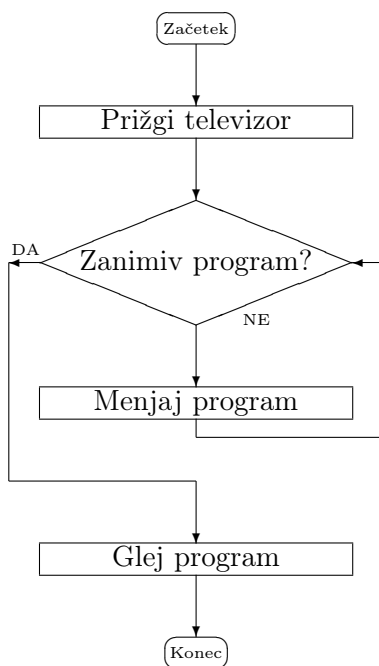
nato ponovimo in ga ponavljamo dokler nismo zadovoljni s slanostjo jedi. Postopek opisuje slika 7.3.



Slika 7.3: Postopek soljenja jedi

*Pri soljenju predvidevamo, da nam je na razpolago dovolj soli.*

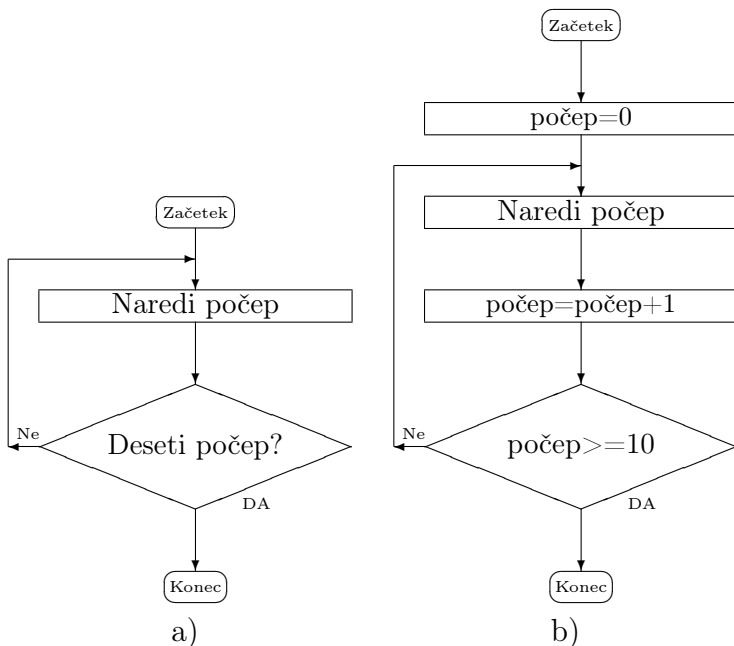
Kot naslednji primer vzemimo iskanje primernega televizijskega programa, pri predpostavki, da ne vemo vnaprej kaj je na sporedu in kaj želimo gledat. Zapisano drugače, vsedemo se pred televizijo in gledamo. Najprej prižgemo televizor in menjavamo programe tako dolgo, dokler se ne ustavimo na programu, ki vzbudi naše zanimanje. Postopek opisuje diagram iz slike 7.6.



Slika 7.4: Postopek menjave programov

Pri menjavi programov predvidevamo, da se uporabnik slej ko prej odloči za en program. Ko pridemo do zadnjega programa, se iskanje nadaljuje na prvem programu (lahko da je sedaj na sporedu druga oddaja).

Kot tretji primer vzemimo primer iz telovadbe. Za nalogo dobimo deset počepov. Počepi delamo dokler jih ne naredimo deset.

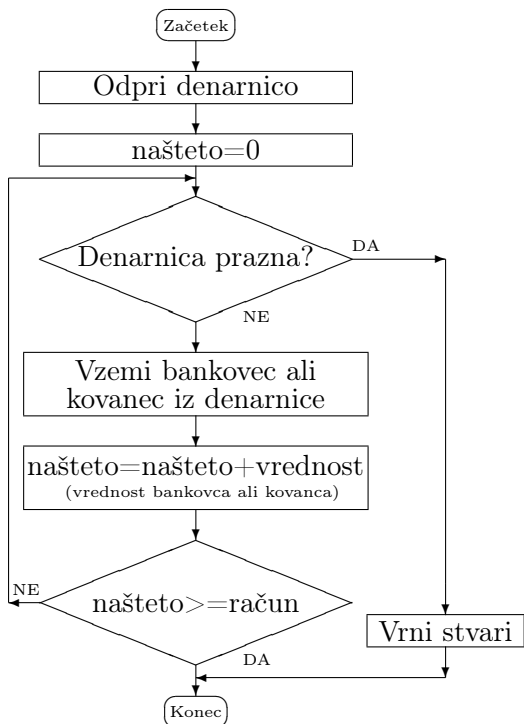


Slika 7.5: Opis ponovitve vaje

Levi diagram iz slike 7.5 opisuje preveč posplošen diagram, saj računalnik ne pozna odgovora na pogoj "Deseti počep?". Za uspešno izvajanje moramo uvesti števec počepov, tj. spremenljivko `počep`, ki ima na začetku vrednost 0, ko ni narejen še noben počep. Po vsakem počepu, povečamo vrednost spremenljivke `počep` za ena. Sedaj se lahko v pogoju sklicujemo na spremenljivko `počep`, kar omogoča, da se odločitev izračuna s pomočjo izraza. Diagram je opisan na desni strani slike 7.5.



Kot nekoliko obsežnejši primer si pogledjmo primer plačevanja računa s pomočjo gotovine. Najprej odpremo denarnico in če ni prazna začnemo naštevati denar, dokler znesek naštetega denarja ni enak ali večji od zneska, ki je na računu.



Slika 7.6: Plačevanje računa

Za uspešno plačevanje računa smo uvedli spremenljivko **našteto**, kjer prištevamo zneske bankovcev ali kovancev, ki smo jih nazadnje vzeli iz denarnice. Postopek končamo, ko naštejemo dovolj denarja ali pa nam je zmanjkalo denarja. V slednjem premeru moramo vrniti stvari, ki smo jih želeli kupit.

## 7.2 Sled izvajanja programa

Zaradi vejitev in ponavljanj je razumevanje programa dodatno oteženo. Pri razumevanju delovanja programa si lahko pomagamo s sledjo izvajanja programa. Sled izvajanja je tabela, kjer beležimo zaporedje klicev ukazov in trenutno stanje spremenljivk. V primeru, da se vrednost spremenljivke ne spremeni, koraka ni potrebno beležiti. Če se pri vejitvah vrednost spremenljivk ne spremeni, jih prav tako ni potrebno beležiti. Vsak klic predstavlja en korak izvajanja programa. V primeru zanke se posamezen ukaz lahko večkrat izvede in je posledično zastopan v več korakih.

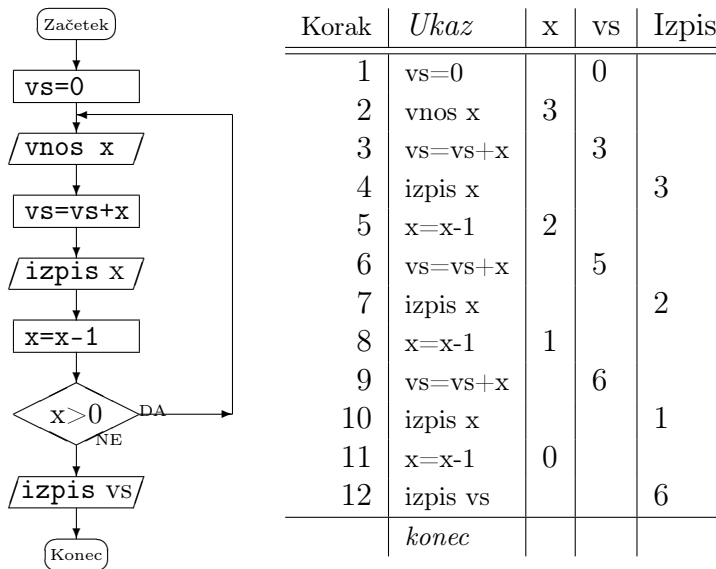


Tabela 7.1: Primer programske sledi (za vhod 3)

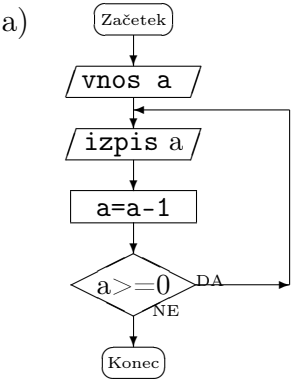
Primer programske sledi za program, ki izračuna vsoto vrste, prikazuje tabela 7.1. Program mora pri vneseni vrednosti 3

izračunati vsoto  $3+2+1$ . Rezultat je torej 6. V glavo tabele programske sledi smo zapisali v prvi stolpec zaporedno številko koraka. V drugi stolpec smo informativno zapisali ukaz, ki se je zgodil v tem koraku (običajno tega stolpca ne pišemo). V tretjem in četrtem stolpcu so spremenljivke, ki nastopajo v programu (v primeru več spremenljivk imamo več stolpcev). Pri spremenljivkah v stolpcih vodimo vrednost spremenljivk v danem koraku. Stolpec izpis nam ponazarja izpis programa na ekran. Opazimo lahko, da program poleg rezultata, sproti izpisuje še vrednost spremenljivke  $x$  (tabela 7.1).

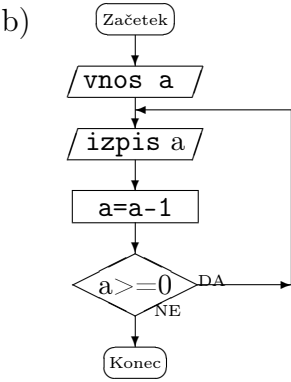
Pri reševanju zahtevnejših nalog svetujemo uporabo programskih sledi. Programsko sled si kot pomoč skicirajte ob diagramu.

### 7.2.1    Dopolni sled

Cilj igre je dopolnit sled tako, da ustreza podanemu diagramu poteka. Sive celice je potrebno zapolniti z manjkajočimi ukazi, vrednostmi spremenljivk ali izpisi.

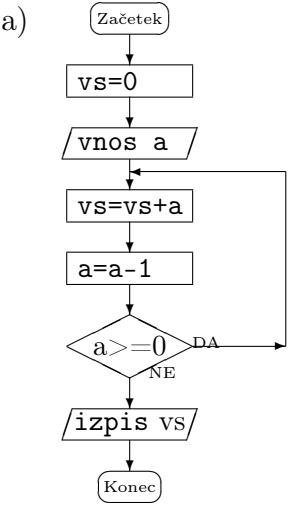


Korak	Ukaz	a	Izpis
1	vnos a	4	
2	izpis a		4
3	a = a - 1	3	
4	izpis a		3
5	a = a - 1	2	
6	izpis a		2
7	a = a - 1	1	
8	izpis a		
9	a = a - 1		
10	izpis a		
11	a = a - 1		
	konec		



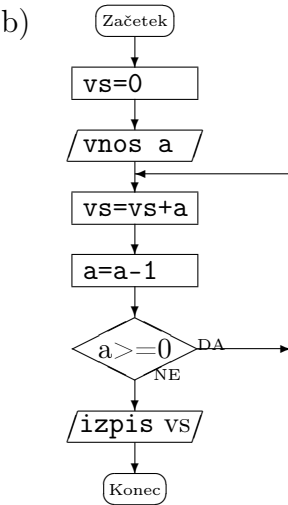
Korak	Ukaz	a	Izpis
1	vnos a	2	

Dopolni sled izvajanja.



★

Korak	Ukaz	vs	a	Izpis
1	vs = 0	0		
2	vnos a		3	
3	vs = vs + a	3		
4	a = a - 1		2	
5	vs = vs + a	5		
6	a = a - 1		1	
7	vs = vs + a			
8	a = a - 1		0	
9	vs = vs + a			
10	a = a - 1		-1	
11	izpis vs			
	konec			

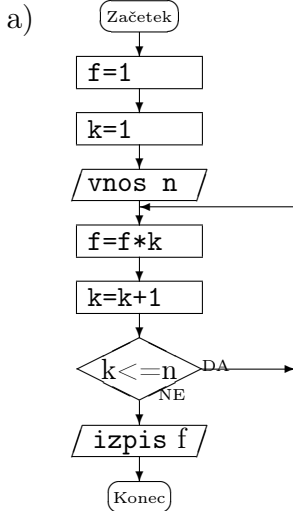


★★

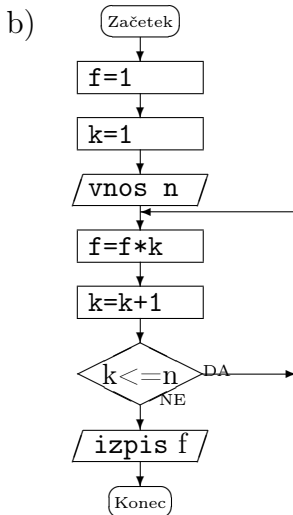
Korak	Ukaz	vs	a	Izpis
1	vs = 0	0		
2	vnos a		2	

Diagram poteka opisuje izračun vsote vrste od a do 0:

$$vs = \sum_{n=0}^a n = 0 + 1 + 2 + \dots + a$$



#	<i>Ukaz</i>	f	k	n	Izpis
1	$f = 1$	1			
2	$k = 1$		1		
3	vnos n			4	
4	$f = f * k$	1			
5	$k = k + 1$				
6	$f = f * k$				
7	$k = k + 1$				
8	$f = f * k$				
9	$k = k + 1$				
10	$f = f * k$				
11	$k = k + 1$				
12	izpis f				
	<i>konec</i>				

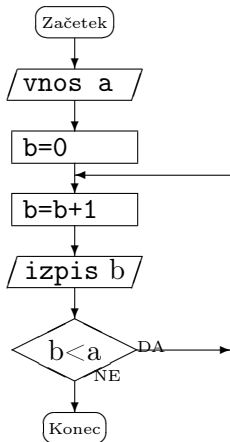
[illegible]

\* Diagram poteka opisuje izračun fakultete od **n**:

$$n! = \prod_{k=1}^n k = 1 * 2 \dots * n$$

### 7.3 Zapiši izpise

Ob podanih vhodnih podatkih (podanih v obliki spodnje tabele) zapiši izpis programa.



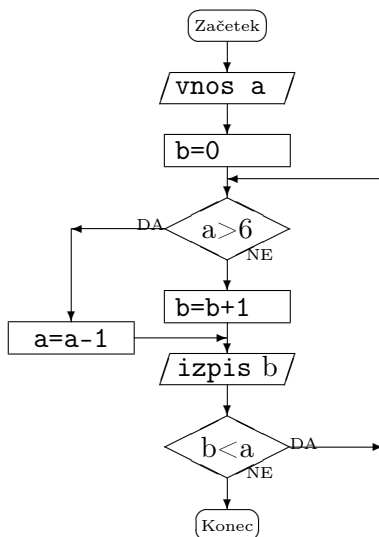
Korak	Ukaz	a	b	Izpis
1	vnos a	3		
2	b=0		0	
3	b=b+1		1	
4	izpis b			1
5	b=b+1		2	
6	izpis b			2
7	b=b+1		3	
8	izpis b			3
konec				

Za primer reševanja naredimo programsko sled pri vnosu vrednosti 3 (korak 1). Spremenljivka *b* se nastavi na 0 (korak 2). Nato se *b* poveča za ena in prvič se izpiše vrednost spremenljivke *b*, tj. 1. (koraka 3 in 4), ker je *b*<*a*, se izvajanje vrne na ukaz *b*=*b*+1 (korak 5), nato se ponovno izpiše vrednost, tj. 2 (korak 6). Ker je dva še vedno manjše od tri, postopek ponovimo in izpiše se 3 (koraka 7 in 8). Ko je vrednost spremenljivke *b* enaka tri, pogoj ne drži več in izvajanje se ustavi. Kot rezultat izpisa bomo navedli 123, tj. izpis brez presledkov in lomljenja vrstic.

	Vnos a	Izpis a	
	3	123	
a)	5		★★
b)	-1		★★
c)	0		★★

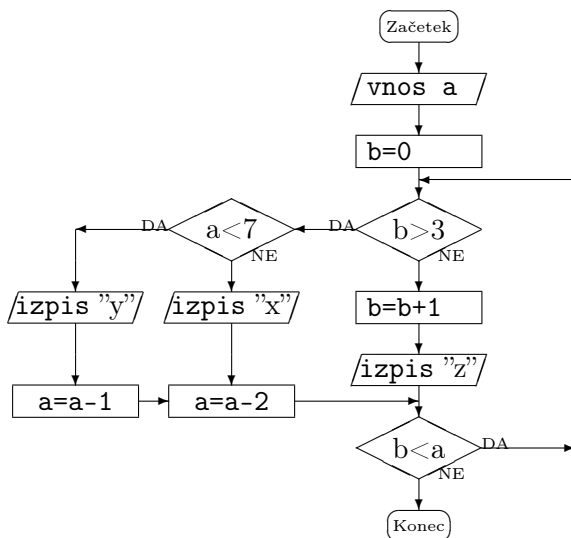


Ob podanih vhodnih podatkih, določi izpis programa.  
Izpis bo potekal brez presledkov in lomljenja vrstic.



	Vnos a	Izpis (vsi izpisi)	
a)	2		★★★
b)	-1		★★★
c)	7		★★★
č)	8		★★★
d)	9		★★★

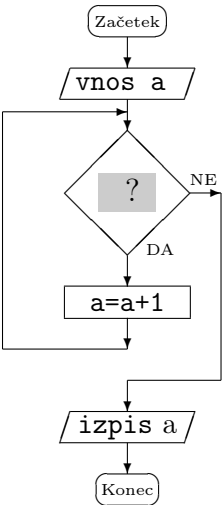
Ob podanih vhodnih podatkih zapiši izpis programa.  
Izpis bo potekal brez presledkov in lomljenja vrstic. V nalogi bomo namesto vrednosti spremenljivk, izpisovali konstante (znake) x, y in z.



	Vnos a	Izpis (vsi izpisi)	
a)	2		★★★
b)	4		★★★
c)	5		★★★
č)	10		★★★
d)	8		★★★

## 7.4 Izberi pogoj

Cilj igre je izbrati pogoj-e v diagramu tako, da dobimo željen izpis. Rezultat je podan v tabeli.



Primer preizkušanja pogojev pri  
vnosu 6 in izpisu 8

Korak	Ukaz	Pogoj	a	Izpis
1	vnos a	$a < 3$	6	
2	izpis a	$a < 3$		6
konec (izpis 6 ne ustreza pogoju)				
1	vnos a	$a > 3$	6	
2	$a = a + 1$	$a > 3$	7	
3	$a = a + 1$	$a > 3$	8	
4	$a = a + 1$	$a > 3$	9	
⋮	⋮	⋮	⋮	
(se nikoli ne konča; ne ustreza pogoju)				

Preveri še ostale pogoje

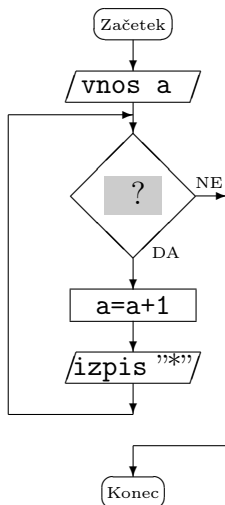
Izberi med naslednjimi pogoji:  $a < 3$ ,  $a > 3$ ,  $a < 8$ ,  $a == 8$  in  $a < > 8$ .

Za primer iščimo pogoj, ki ob vnosu števila 6 izpiše število 8. Pravilnost lahko preverjamo s sledjo, za vsak pogoj (tabela zgoraj) ali pa na pamet.

Med ponujenimi pogoji ustrezata vhodu in izhodu samo  $a < > 8$  in  $a < 8$  (spodaj že izpolnjeno).

	Vnos a	Izpis	Pogoj	
	6	8	$a < > 8$ ali $a < 8$	
a)	8	9		★★★
b)	3	8		★★★
c)	0	3		★★★

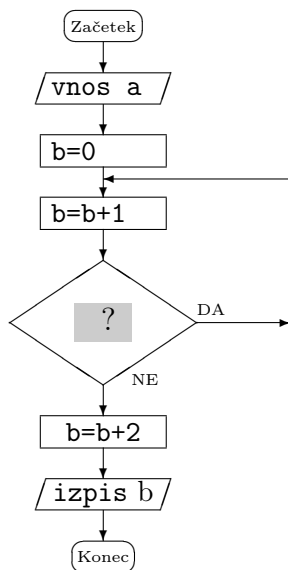
Izberi pogoj-e v diagramu tako, da dobimo željen izpis. Kot rezultat bomo tokrat izpisovali zvezdice. Število zvezdic v tabeli ponazarja število ponovitev zanke.



Izberi med naslednjimi pogoji:  $a < 3$ ,  $a > 3$ ,  $a < 8$ ,  $a == 8$  in  $a < > 8$ .

	Vnos a	Izpis	Pogoj	
a)	2	*		★★★
b)	1	**		★★★
c)	6	**		★★★
č)	0	*****		★★★
d)	8	*		★★★
e)	4	****		★★★

Izberi pogoj-e v diagramu tako, da dobimo željen izpis. Rezultat je podan v tabeli. Bodite pozorni, da imamo tokrat dve spremenljivki, a in b (v prejšnji nalogi smo imeli samo a).



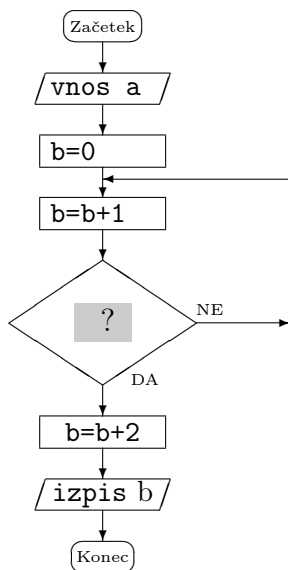
Izberi med naslednjimi pogoji:  $b>3$ ,  $b<4$ ,  $b==1$ ,  $a>6$  in  $a<>b$ . Za primer iščimo pogoj, ki ob vnosu števila 5 izpiše število 7.

Vnos a	Izpis b	Pogoj
5	7	?

Pravilen odgovor je  $a<>b$  (spodaj že izpolnjen).

	Vnos a	Izpis b	Pogoj	
	5	7	$a<>b$	
a)	0	3		★★★
b)	2	6		★★★
c)	5	3		★★★

Izberi pogoj-e v diagramu tako, da dobimo željen izpis. Diagram je identičen prejšnjemu, samo da sta pri pogoju zamenjana DA in NE.



Izberi med naslednjimi pogoji:  $b > 3$ ,  $b < 4$ ,  $b == 1$ ,  $a > 6$  in  $a < b$ .

	Vnos a	Izpis b	Pogoj	
a)	0	3		★★★
b)	2	6		★★★
c)	5	3		★★★
č)	7	3		★★★
d)	10	6		★★★

## 7.5 Poveži

S puščicami poveži simbole diagrama poteka tako, da bo program ob podanih vseh izpisal pričakovan rezultat. Vhod in izpis sta podana v obliki tabele. Vsaka vrstica v tabeli predstavlja eden izmed možnih rezultatov, oz. zagonov programa. Pri povezovanju uporabite tudi zanke (povratne puščice). Izpis v tabeli je brez presledkov in lomljenih vrstic.

Vnos a	Izpis
3	456789
7	89
8	9
9	brez

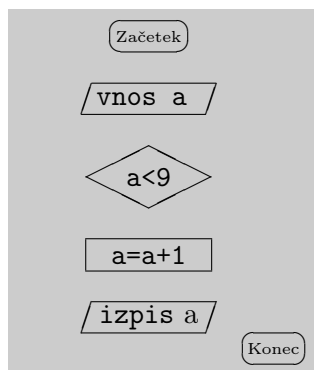
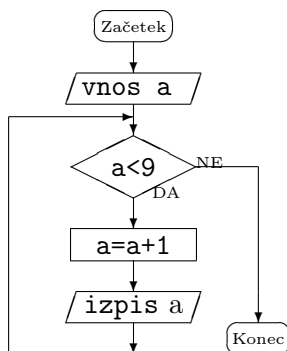


Diagram je pravilno povezan takrat, kadar ustreza vsem vrsticam v podani tabeli (črne celice). Pravilno povezan diagram, za podano tabelo, je prikazan na spodnji sliki. Pri povezovanju ne pozabit označiti pogojev z DA in NE in na pravila povezovanja (stran 81).

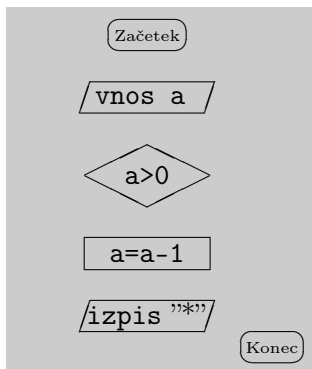


S puščicami poveži simbole diagrama tako, da bo program ob podanih vseh izpisal pričakovan rezultat.

★★★

a)

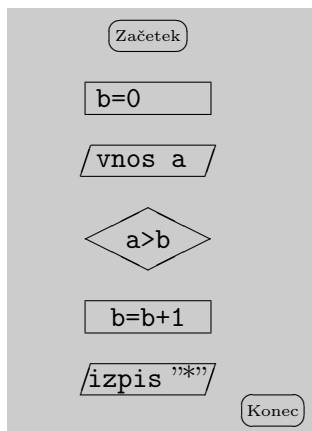
Vnos	Izpis
1	*
3	***
8	*****
0	brez



★★★

b)

Vnos	Izpis
3	***
7	*****
1	*
0	brez



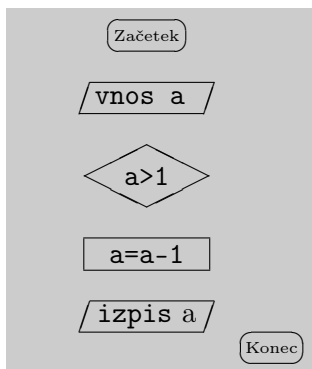
**Namig:** Pred povezovanjem poizkusite ugotoviti, kakšna je povezava med vhodno vrednostjo in dolžino izpisa, tj. različnih izpisov. Podatek nam nakazuje število ponavljanj.



S puščicami poveži simbole diagrama, da bo program ob podanih vseh vhodih izpisal pričakovan rezultat. Pri povezovanju uporabite tudi zanke (povratne puščice).

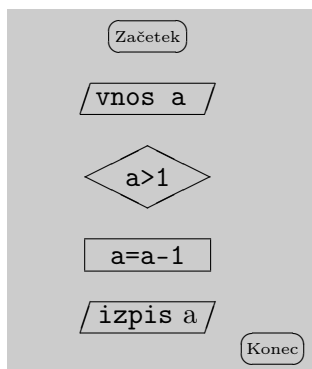
★★★

Vnos	Izpis
3	21
7	654321
8	7654321
a)	1
	brez



★★★

Vnos	Izpis
3	32
7	765432
8	8765432
b)	1
	1



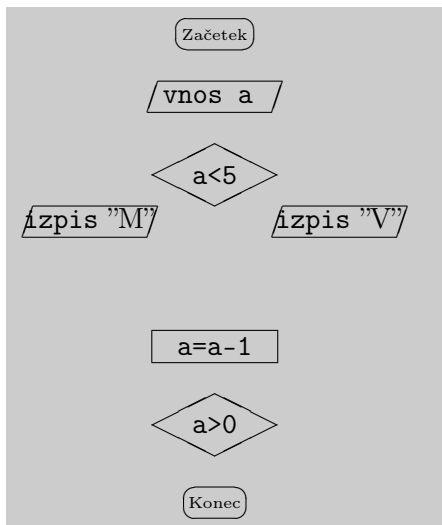
**Namig:** Preučite vrstni red izvajanja ukazov, tj. ali se prvo izvede ukaz in nato pogoj ali obratno.

S puščicami poveži simbole diagrama, da bo program ob podanih vseh vhodih izpisal pričakovan rezultat. Pri povezovanju uporabite tudi zanke (povratne puščice).

★★★★

a)

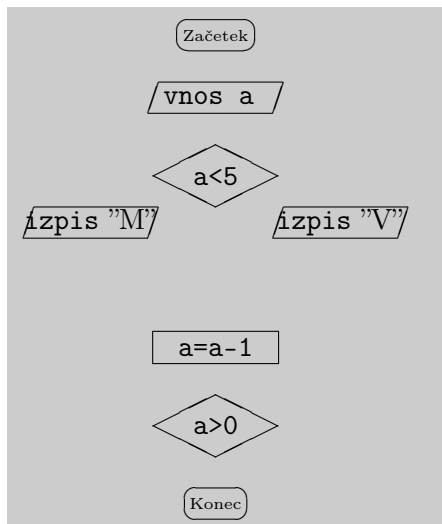
Vnos	Izpis
1	M
5	VMMMM
7	VVVMMMM
0	M
2	MM



★★★★

b)

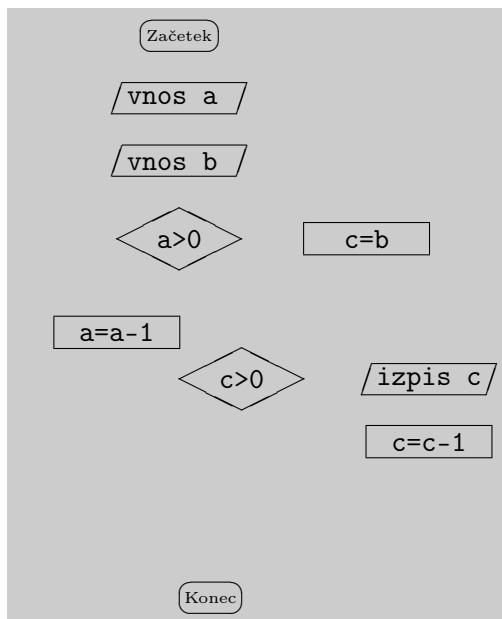
Vnos	Izpis
1	V
5	MVVVV
7	MMMVVVV
0	brez
2	VV



S puščicami poveži simbole diagrama tako, tako da bo program ob podanih vseh izpisal željen rezultat. Program prebere spremenljivki a in b, ter ju na koncu izpiše. Uporabite zanke.



Vnos		Izpis
a	b	
2	3	321321
3	2	212121
4	1	1111
2	4	43214321



Več nalog različnih težavnostnih stopenj lahko najdete v prilogi na straneh od 176 do 187.

## 7.6 Primeri iz prakse

Zmožnost ponavljanja je ena izmed najpomembnejših lastnosti računalnika. Računalnik lahko ponovi ukaz ali zaporedje ukazov tudi več milijonkrat na sekundo, pri tem pa ne naredi nobene napake in se nič ne utruji. Pri programiranju ukaze za ponavljanje imenujemo zanke. Pri večini splošno namenskih programskih jezikov srečamo tri tipe ukazov za opis zank, ki jih opisuje slika 7.2.

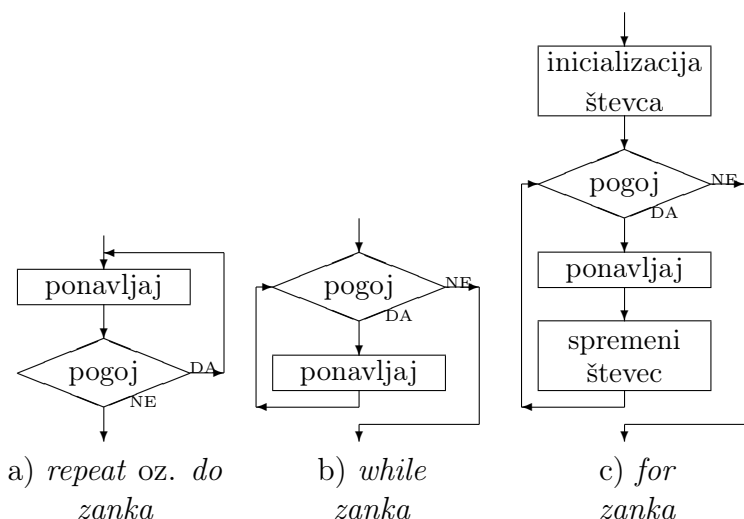


Tabela 7.2: Različni tipi zank

Glavna razlika med prvim in drugim tipom zank (slika 7.2) je v tem, da se pri prvem tipu akcija "ponavljaaj" izvrši v vsakem primeru vsaj enkrat. Tipa zank dva in tri sta si po osnovi

podobna. Glavna razlika je v tem, da se zaradi pogostosti uporabe števecv v zankah uporablja nov tip zanke, ki vsebuje akciji inicializacije in spremembe števca že v sintaksi zanke (*for zanka*). Tretji tip zanke lahko brez težav zapišemo s pomočjo drugega tipa. Informativno si pogledjmo primere zank v različnih programskih jezikih.

Tip zanke	Primer
1	<b>repeat</b> akcija ; <b>until</b> pogoj;
2	<b>while</b> pogoj akcija ;
3	<b>for</b> x := 1 <b>to</b> 10 <b>do</b> akcija ;

Tabela 7.3: Pimeri zank v programskem jeziku Pascal

Tip zanke	Primer
1	<b>do</b> { akcija ; } <b>while</b> ( pogoj );
2	<b>while</b> (pogoj) akcija ;
3	<b>for</b> (int x=1; x<10; x=x+1) akcija ;

Tabela 7.4: Primeri zank v prog. jezikih Java, C++ in C#

Programski jeziki Java, C++ in C#, imajo korenine v programskem jeziku C, zato je njihova sintaksa pri zankah skoraj identična.

Tip zanke	Primer
1	<b>loop do</b> akcija <b>break unless</b> pogoj <b>end</b>
2	<b>while</b> pogoj akcija <b>end</b>
3	<b>for</b> x <b>in</b> 1..10 akcija <b>end</b>

Tabela 7.5: Primeri zank v programskem jeziku Ruby

V poglavjih "Primeri iz prakse", smo do sedaj prikazovali samo primere ukazov. V naslednjem poglavju pa bomo prikazali primer celega programa zapisanega v različnih programskih jezikih.

# 8.

*Teorija brez prakse je kot voz brez kolesa.*

*- Latinski pregovor -*

## Primeri iz prakse

### Znano

- ★ Stavek, ki ima enak pomen se lahko v različnih jezikih, pove-zapiše različno.
- ★ Poglavje je informativno in ne vsebuje veliko podrobnosti o posameznih programskih jezikih.
- ★ Programski jezik ne dovoljuje slovničnih napak.

V svetu trenutno poznamo več tisoč programskih jezikov. V knjigi smo se omejili na nekaj razširjenjih, splošno namenskih programskih jezikov. Končni cilj programiranja je, da nek postopek avtomatiziramo, tako, da se izvaja na računalniku. Postopek lahko na grobo razdelimo na tri korake, ki pa se v praksi med seboj prepletajo:

1. Najprej moramo vedeti, kaj naj program počne (analiza domene) in ga definirati v obliki postopka. V tem koraku imata največ dela naročnik in načrtovalec programa.
2. Zapisati postopek v izbranem programskem jeziku (program).

3. Uporabljati program. V prvi fazi je to delo testerja (preverja pravilnost delovanja programa), nato pa naročnika oz. končnega uporabnika.

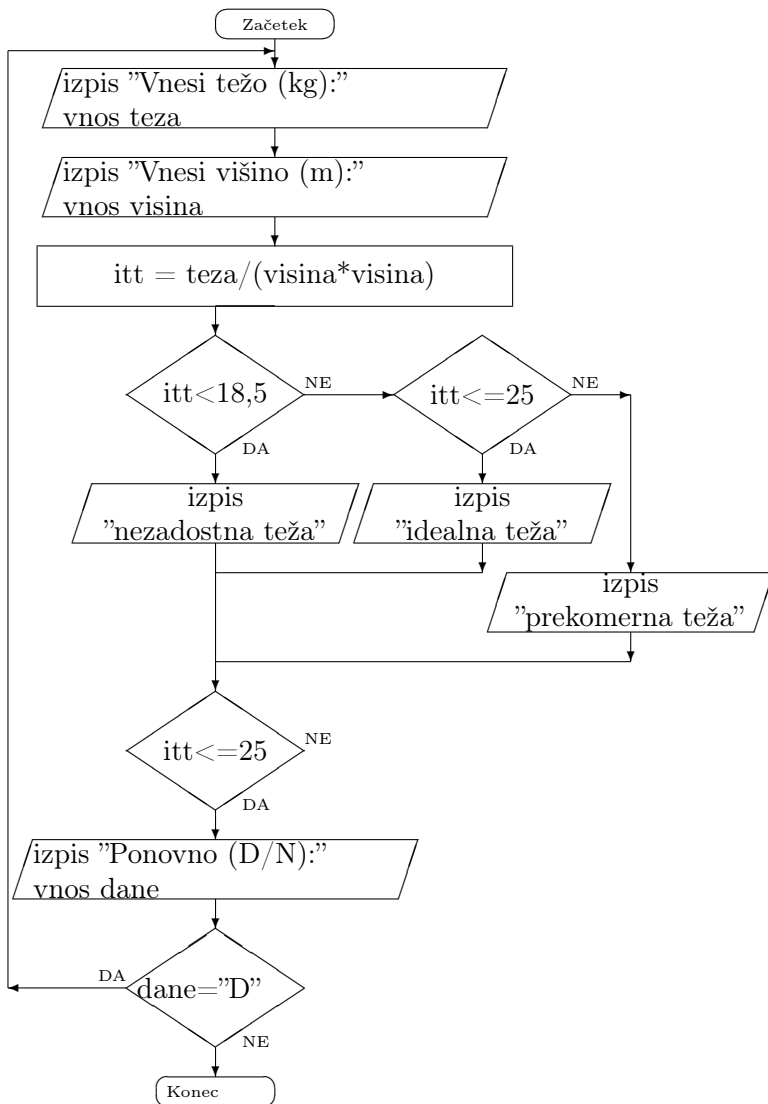
Posamezni programski jeziki imajo lahko veliko specifik, ki pa jih v tej knjigi ne bomo obravnavali. V poglavju bomo bralcu prikazali celoten program v različnih programskih jezikih, ki daje bralcu vpogled v razlike med posameznimi programskimi jeziki. Sintaksa programskih jezikov, tj. pravil za vrstni red besed, sintaktičnih elementov in njihovih parametrov, se lahko zelo razlikuje od jezika do jezika. Zato je za samostojno programiranje potrebna literatura za izbran programski jezik.

Primeri prav tako vsebujejo veliko dodatnih ukazov, ki so del knjižnic in jih potrebujemo za delo z vhodno izhodnimi napravami, tj. v našem primeru branje iz tipkovnice in pisanje na ekran.



## 8.1 Zahteve programa

Poglejmo si sedaj problem, ki ga bomo rešili v različnih programskih jezikih. Cilj programa je izračun indeksa telesne teže. Program naj omogoča izračun za eno ali več oseb.



*progo.crepinsek@gmail.com*

---

## 8.2 Pascal

Programski jezik Pascal je razvil Niklaus Wirth, leta 1970. Cilj je bil spodbujanje dobrih praks strukturnega programiranja, kasneje so jezik razširi z objektnimi koncepti. Danes se programski jezik uporablja predvsem v izobraževalne namene.

```
1 program IzracunIndeksaTelesneTeze;
2   var teza, visina, itt:real;
3       dane:char;
4 begin
5   repeat
6     write('Vnesi težo (kg):');
7     readln(teza);
8     write('Vnesi višino (m):');
9     readln(visina);
10    itt := teza/(visina*visina);
11    if itt < 18.5 then
12      writeln('nezadostna teža')
13    else
14      if itt <= 25 then
15        writeln('idealna teža')
16      else
17        writeln('prekomerna teža');
18    write('Ponovno (D/N)');
19    readln(dane);
20    until dane <> 'D';
21 end.
```

Omenimo nekaj značilnosti programa. Vse spremenljivke je potrebno deklarirati v glavi programa ali procedure (vrstici 2 in 3). Zanka "repeat-until" se izvaja dokler pogoj ni izpolnjen (vrstica 20). Ob koncu vsakega ukaza zapišemo podpičje. Sestavljen stavek, tj. stavek ki ima več stavkov, je obdan z rezerviranimi besedama **begin** in **end**.

## 8.3 Java

Programski jezik Java je razvilo podjetje Sun Microsystems, leta 1995. Jezik je nastal iz potrebe po zaneslivejšem in preprostejšem programskem jeziku, ki bi služil kot osnova javanski platformi, katere cilj je prenosljivost izvršne kode (uporaba vmesne kode). Osnovo so povzete po programskih jezikih C in C++. Java trenutno velja za enega izmed najpopularnejših programskih jezikov.

```
1 public class IzracunIndeksaTelesneTeze {
2     public static void main(String [] args)
3         throws Exception {
4         BufferedReader in = new BufferedReader(
5             new InputStreamReader(System.in));
6         double teza; //v kg
7         double visina; //v metrih
8         double itt; //izračunana
9         String dane; //za ponovitev vnosa
10        do {
11            System.out.print("Vnesi težo (kg):");
12            teza = Double.parseDouble(in.readLine());
13            System.out.print("Vnesi višino (m):");
14            visina = Double.parseDouble(in.readLine());
15            itt = teza/(visina*visina);
16            if (itt < 18.5) {
17                System.out.println("nezadostna teža");
18            } else
19                if (itt <= 25) {
20                    System.out.println("idealna teža");
21                } else
22                    System.out.println("prekomerna teža");
23            System.out.println("Ponovno (D/N):");
24            dane=in.readLine();
25        }
26        while (dane.equals("D"));
27    }
28 }
```

## 8.4 C#

Programski jezik C# je razvilo podjetje Microsoft leta 2000, kot odgovor na programski jezik Java. Cilj jezik C# je biti preprost, moderen, splošno namenski, objektno orientiran programski jezik.

```
1 using System.Text;
2 using System.IO;
3
4 namespace TelesnaTeza {
5     class IzracunIndeksaTelesneTeze {
6         static void Main(string[] args) {
7             double teza; //v kg
8             double visina; //v metrih
9             double itt; //izracunana
10            string dane; //za ponovitev vnosa
11            try {
12                do {
13                    Console.Write("Vnesi težo (kg):");
14                    teza = Double.Parse(Console.ReadLine());
15                    Console.Write("Vnesi višino (m):");
16                    visina = Double.Parse(Console.ReadLine());
17                    itt = teza / (visina * visina);
18                    if (itt < 18.5)
19                        Console.WriteLine("nezadostna teža");
20                    else if (itt <= 25)
21                        Console.WriteLine("idealna teža");
22                    else
23                        Console.WriteLine("prekomerna teža");
24                    Console.WriteLine("Ponovno (D/N):");
25                    dane = Console.ReadLine();
26                } while (dane.Equals("D"));
27            } catch (Exception e) {
28                Console.WriteLine("Exception");
29            }
30        }
31    }
32 }
```

## 8.5 C++

Programski jezik C++ je napisal dr. Bjarne Stroustrup, leta 1979, v podjetju Bell Labs, kot izboljšavo programskega jezika C. Programska jezika C in C++ sta dolga leta veljala kot najpopularnejša programska jezika. Programski jezik C++ podpira veliko konceptov, ki pa so se v praksi pokazali za zapletene in so botrovali k nepravilni uporabi, s tem pa pripomogli k skritim napakam v programski opremi.

```
1 #include <iostream.h>
2 int main()
3 {
4     double teza; //v kg
5     double visina; //v metrih
6     double itt; //izračunana
7     char dane; //za ponovitev vnosa
8     do {
9         cout<<"Vnesi težo (kg): ";
10        cin>>teza;
11        cout<<"Vnesi višino (m): ";
12        cin>>visina;
13        itt = teza/(visina*visina);
14        if (itt < 18.5) {
15            cout<<"nezadostna teža"<<endl;
16        } else
17            if (itt <= 25) {
18                cout<<"idealna teža"<<endl;
19            } else
20                cout<<"prekomerna teža"<<endl;
21        cout<<"Ponovno (D/N): ";
22        cin>>dane;
23    }
24    while (dane == 'D');
25 }
```

## 8.6 Ruby

Programski jezik Ruby je razvil japonec Yukihiro "Matz" Matsumoto, leta 1995. Ruby je objektno orientiran programski jezik, katerega cilj je biti programerju prijazen programski jezik. Ruby velja za programski jezik v vzponu. Njegova glavna slabost je, da deluje kot interpreter, torej program se ne prevede direktno v izvršno kodo.

```
1 loop do
2   puts 'Vnesi težo (kg): '
3   teza = gets.chomp.to_f
4   puts 'Vnesi višino (m): '
5   visina = gets.chomp.to_f
6   itt = teza / (visina * visina)
7   if itt < 18.5
8     puts 'nezadostna teža'
9   else
10    if itt <= 25
11      puts 'idealna teža'
12    else
13      puts 'prekomerna teža'
14    end
15  end
16  puts 'Ponovno (D/N): '
17  dane = gets.chomp
18  break unless dane == 'D'
19 end
```

## 8.7 Ostalo

Izbira programskega jezika je stvar namena in osebne presoje. Pri izbiri moramo biti pozorni na podporo programskemu jeziku, to pomeni kakšna orodja za razvoj so na voljo, literatura, pomoč različnih skupin na internetu, popularnost v okolju v katerem delujete, itd.

*progo.crepinsek@gmail.com*

---



# 9.

*Glej daleč, in ko misliš, da  
že gledaš deleč, glej še dlje.*

*- Baden Powel -*

## Kako dalje

### Znano

- ★ Vaja dela mojstra.
- ★ Vsaka knjiga ima svoj začetek in konec.
- ★ To je zadnje poglavje.

Prišli ste, do zadnjega poglavja. Za uspešno razmišljanje o algoritmih je potrebno veliko vaj, zato vsebuje knjiga priložilo, kjer lahko najdete še veliko nalog iz vsakega poglavja v tej knjigi. Po uspešnem "treningu" s to vadnico, lahko nadaljujete "trening" na spletnem naslovu: [www.progo?.](http://www.progo?)

Če se odločite za programiranje na računalnik, lahko najdete več informacij o posameznem programskem jeziku na naslednjih spletnih naslovih:

**Pascal** <http://www.freepascal.org>

**C++** <http://www.bloodshed.net/devcpp.html>

**Java** <http://java.sun.com>

**C#** <http://msdn.microsoft.com/vcsharp>

*progo.crepinsek@gmail.com*

---

**Ruby** <http://www.ruby-lang.org>

# 10.

*Ponavljanje je mati študija.*

*- Latinski pregovor -*

## Priloga

Priloga vsebuje dodatne naloge, ki so razdeljene po poglavjih, v katerih so zapisana natančnejša navodila. Naloge so označene s težavnostjo od ene do pet zvezdic.

## 10.1 Izpis in konstante

### 10.1.1 Določi vrstni red konstant

V sive celice zapišite pravilni vrstni red konstant x, y, z, w, k in i.

```
x = "ab"
y = "bca"
z = "cab"
w = "ac"
k = "bc"
i = "acb"
```

- a) 

	1	2	3	4	5	6	7	8
	acababab							
<hr/>								
<hr/>								
izpis	<div></div>							

 ★
- b) 

	1	2	3	4	5	6	7	8	9
	acbacacab								
<hr/>									
<hr/>									
izpis	<div></div>								

 ★
- c) 

	1	2	3	4	5	6	7	8	9	10
	cabbcabcab									
<hr/>										
<hr/>										
izpis	<div></div>									

 ★
- č) 

	1	2	3	4	5	6	7	8	9	10	11	12
	cabbcacabbca											
<hr/>												
<hr/>												
izpis	<div></div>											

 ★

V sive celice zapišite možne rešitve za konstante x, y, z, w, k in i.

```
x = "ab"
y = "bca"
z = "cab"
w = "ac"
k = "bc"
i = "acb"
```

- a) 

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
	a	c	b	c	a	b	a	c	b	b	c	a	c	b
<hr/>														
izpis	<div></div>													

★★
----
- b) 

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	b	c	b	c	a	a	c	c	a	b	b	c	c	a	b
<hr/>															
izpis	<div></div>														

★★
----
- c) 

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	b	c	b	c	b	c	a	b	c	a	b	c	a	a	c
<hr/>															
izpis	<div></div>														

★★★★
------
- č) 

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
	b	c	c	a	b	a	b	a	b	a	c	b	c	a	b	c	a	b
<hr/>																		
izpis	<div></div>																	

★★★★
------

V sive celice zapišite pravilni vrstni red konstant a, b in c.

a = "bc" ali a = "caba"

b = "ca"

c = "ba"

★★★★

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20  
cabcbabccabcbabccaba

a) izpis<sub>1</sub>   
izpis<sub>2</sub>   
izpis<sub>3</sub>

★★★★

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24  
bacabababccabcbabccacaba

b) izpis<sub>1</sub>   
izpis<sub>2</sub>   
izpis<sub>3</sub>   
izpis<sub>4</sub>

★★★★

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26  
babccabacabccababacabacaba

c) izpis<sub>1</sub>   
izpis<sub>2</sub>   
izpis<sub>3</sub>   
izpis<sub>4</sub>

V sive celice zapišite pravilni vrstni red konstant a, r, b in n.

a = "ra" ali a = "ara"

r = "ba"

b = "na"

n = "bana"

★★★

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

naarabanaararabara

a) izpis<sub>1</sub>

izpis<sub>2</sub>

izpis<sub>3</sub>

b

★★★

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

narabanaaraarabaara

b) izpis<sub>1</sub>

izpis<sub>2</sub>

izpis<sub>3</sub>

b

c)

★★★★

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27

banaarabaaranarabanarabaara

izpis<sub>1</sub>

izpis<sub>2</sub>

izpis<sub>3</sub>

n

*progo.crepinsek@gmail.com*

---

V sive celice zapišite pravilni vrstni red konstant  $a$ ,  $s$ ,  $r$  in  $d$ .



```
a = "ar"  ali  a = "da"
s = "sa"
r = "ra"
d = "rasa"
```

	<div> <div>1 2 3 4 5 6 7 8 9 10 11 12</div> <div>raarsaardara</div> </div>	
a)	<div> <div>izpis<sub>1</sub></div> <div>izpis<sub>2</sub></div> <div>izpis<sub>3</sub></div> </div>	★★
	<div> <div>1 2 3 4 5 6 7 8 9 10 11 12 13 14</div> <div>raarsadarasaar</div> </div>	
b)	<div> <div>izpis<sub>1</sub></div> <div>izpis<sub>2</sub></div> <div>izpis<sub>3</sub></div> </div>	★★
	<div> <div>1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16</div> <div>raararrasadaarra</div> </div>	
c)	<div> <div>izpis<sub>1</sub></div> <div>izpis<sub>2</sub></div> <div>izpis<sub>3</sub></div> </div>	★★
	<div> <div>1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20</div> <div>raarrasaarsaarrasada</div> </div>	
č)	<div> <div>izpis<sub>1</sub></div> <div>izpis<sub>2</sub></div> <div>izpis<sub>3</sub></div> </div>	★★★

V sive celice zapišite pravilni vrstni red konstant a, t, c in o.

a = "tata" ali a = "co"

t = "ta"

c = "to"

o = "oc"

★★★★

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18  
octota octata ta octo

a) izpis<sub>1</sub>

izpis<sub>2</sub>

izpis<sub>3</sub>

izpis<sub>4</sub>

★★★★

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22  
tacota tata ta co octota oc

b) izpis<sub>1</sub>

izpis<sub>2</sub>

izpis<sub>3</sub>

izpis<sub>4</sub>

★★★★★

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24  
tatata tata ta ta co octota oc

c) izpis<sub>1</sub>

izpis<sub>2</sub>

izpis<sub>3</sub>

izpis<sub>4</sub>

V sive celice zapišite pravilni vrstni red konstant  $a$ ,  $m$  in  $t$ .

a = "ma" ali a = "ta"  
m = "tama"  
t = "ata"

1 2 3 4 5 6 7 8 9 10 11 12 13  
taataamatamama

★★★★

a) izpis<sub>1</sub>   
izpis<sub>2</sub>   
izpis<sub>3</sub>

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16  
atatamatataatama

★★★★

b) izpis<sub>1</sub>   
izpis<sub>2</sub>   
izpis<sub>3</sub>

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19  
tamataamaatatamama

★★★★

c) izpis<sub>1</sub>   
izpis<sub>2</sub>   
izpis<sub>3</sub>

č)

★★★★★

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28  
maataamaataamatamaatatata

izpis<sub>1</sub>   
izpis<sub>2</sub>   
izpis<sub>3</sub>

V sive celice zapišite pravilni vrstni red konstant  $a$ ,  $r$  in  $k$ .

a = "ka" ali a = "raka"  
r = "kak" ali r = "ra"  
k = "ar"

1 2 3 4 5 6 7 8 9 10 11 12  
arrakaararka

★★★

a) izpis<sub>1</sub>   
izpis<sub>2</sub>   
izpis<sub>3</sub>

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18  
rarakaarrakararaka

★★★★

b) izpis<sub>1</sub>   
izpis<sub>2</sub>   
izpis<sub>3</sub>

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26  
rarakaarkarakakakkakkaarka

★★★★★

c) izpis<sub>1</sub>   
izpis<sub>2</sub>   
izpis<sub>3</sub>

č)

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27  
arkaarrakaarrakararakakarka

★★★★★

izpis<sub>1</sub>   
izpis<sub>2</sub>   
izpis<sub>3</sub>

V sive celice zapišite pravilni vrstni red konstant p in i.

p = "pi" ali p = "ip"  
i = "ii" ali i = "ppp"

★★★★

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
	p	i	p	p	i	i	i	i	p	i	p	p	p	i	i	i	i		
izpis <sub>1</sub>																			
a) izpis <sub>2</sub>																			
izpis <sub>3</sub>																			
izpis <sub>4</sub>	p																		

★★★★

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
	i	i	p	p	p	p	p	p	p	p	p	p	i	p	i	p	i	p	i	i	p	p	i
izpis <sub>1</sub>																							
b) izpis <sub>2</sub>																							
izpis <sub>3</sub>																							
izpis <sub>4</sub>	i																						

★★★★★

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
	i	p	p	p	p	p	i	p	p	p	p	i	i	p	i	p	i	p	i	p	i	p	i	p	i	p
izpis <sub>1</sub>																										
c) izpis <sub>2</sub>																										
izpis <sub>3</sub>																										
izpis <sub>4</sub>	i																									

V sive celice zapišite pravilni vrstni red konstant c, o in p.

c = "co" ali c = "op"  
o = "po" ali o = "oc"  
p = "opo" ali p = "oco"

★★★★

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
	p o o c c o o c o p o p o c c o o p o c																	
a)	izpis <sub>1</sub> <input type="text"/>																	
	izpis <sub>2</sub> <input type="text"/>																	
	izpis <sub>3</sub> <input type="text"/>																	
	izpis <sub>4</sub> <input type="text" value="c"/>																	

★★★★

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	o c o p o o c c o p o p o c c o o c p o c o																			
b)	izpis <sub>1</sub> <input type="text"/>																			
	izpis <sub>2</sub> <input type="text"/>																			
	izpis <sub>3</sub> <input type="text"/>																			
	izpis <sub>4</sub> <input type="text" value="o"/>																			

★★★★★

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
	o c o p o o c c c o o c p o c c o o c c o p o o c o p o																									
c)	izpis <sub>1</sub> <input type="text"/>																									
	izpis <sub>2</sub> <input type="text"/>																									
	izpis <sub>3</sub> <input type="text"/>																									
	izpis <sub>4</sub> <input type="text" value="o"/>																									



V sive celice zapišite pravilni vrstni red konstant a, b in c.

a = "bc" ali a = "caba"

b = "ca" ali b = "babc"

c = "ba"



a)

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44  
cacababcbacababacacababababccabacababc

izpis <sub>1</sub>	
izpis <sub>2</sub>	
izpis <sub>3</sub>	
izpis <sub>4</sub>	c

## 10.1.2 Določi vrednosti konstant

Določi vrednost konstantam.

a)  $x =$

★

```
      1 2 3 4 5 6
      cdccdc
=====
izpis 
```

b)  $x =$    
 $y =$

★★

```
      1 2 3 4 5 6
      cddccd
=====
izpis 
```

c)  $x =$    
 $y =$

★★

```
      1 2 3 4 5 6 7
      kkakkak
=====
izpis 
```

č)  $x =$    
 $y =$

★★★

```
      1 2 3 4 5 6
      aaaaba
=====
izpis 
```

Določi vrednosti konstant  $x$ ,  $y$  in  $z$ .

$x =$    
 a)  $y =$    
 $z =$

★★

```

      1 2 3 4 5 6 7 8
      01230110
=====
izpis  x z x y
      1 2 3 4 5 6 7 8
      01012301
=====
izpis  x x z x
    
```

$x =$    
 b)  $y =$    
 $z =$

★★★

```

      1 2 3 4 5 6
      23212 3
=====
izpis  y x y z x
      1 2 3 4 5 6
      12232 3
=====
izpis  z y x y x
    
```

Določi vrednosti konstant  $x$ ,  $y$  in  $z$ .

$x =$   ★★★★

a)  $y =$

$z =$

	1	2	3	4	5	6	7	8	9	10
	1	1	1	1	1	1	1	1	1	1

---

izpis x y y x

	1	2	3	4	5	6	7
	0	1	1	1	1	1	0

---

izpis z y x z

	1	2	3	4	5	6
	1	1	1	0	1	1

---

izpis x z y

★★★★

$x =$

b)  $y =$

$z =$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
	0	0	0	0	1	0	1	1	0	1	0	0	0	0

---

izpis y x x z z y

	1	2	3	4	5	6	7	8	9	10	11	12	13
	0	1	1	0	1	0	0	1	1	0	0	0	0

---

izpis x z z x z y

	1	2	3	4	5	6	7	8	9	10	11	12	13
	0	1	1	0	0	1	0	0	0	0	1	1	0

---

izpis x z x y x z

Določi vrednosti konstant x, y, z in w.

a)

x =	<input type="text"/>	
y =	<input type="text"/>	
z =	<input type="text"/>	
w =	<input type="text"/>	

★★★★

	1	2	3	4	5	6
	x	y	y	x	y	x

---

izpis z w z y

	1	2	3	4	5	6	7	8
	x	y	y	x	x	y	x	y

---

izpis x y x x

	1	2	3	4	5	6	7	8	9
	y	y	x	y	x	y	x	y	x

---

izpis w x z y y

b)

x =	<input type="text"/>	
y =	<input type="text"/>	
z =	<input type="text"/>	
w =	<input type="text"/>	

★★★★

	1	2	3	4	5	6	7	8	9
	x	y	y	x	x	x	y	x	x

---

izpis z w x z y x

	1	2	3	4	5	6
	x	y	y	x	x	x

---

izpis z w z x

	1	2	3	4	5	6	7	8
	y	y	y	x	x	x	y	x

---

izpis w y z x y z

## 10.2 Vnos in spremenljivke

### 10.2.1 Prečrti odvečne vrstice

Če obstajajo odvečne vrstice jih prečrti, in v sive celice zapišite izpis programa.

	<div>#</div>	<div>Program</div>	<div>Izpis</div>	<div>★★</div>
	<div>1</div>	<div>x = 7</div>		
a)	<div>2</div>	<div>x = 9</div>		
	<div>3</div>	<div>x = x + 1</div>		
	<div>4</div>	<div>izpis x</div>	<div></div>	
	<div>#</div>	<div>Program</div>	<div>Izpis</div>	<div>★★</div>
	<div>1</div>	<div>x = 7 + 3</div>		
b)	<div>2</div>	<div>y = x</div>		
	<div>3</div>	<div>z = x</div>		
	<div>4</div>	<div>z = 1</div>		
	<div>5</div>	<div>izpis z</div>	<div></div>	
	<div>#</div>	<div>Program</div>	<div>Izpis</div>	<div>★★</div>
	<div>1</div>	<div>x = 3</div>		
c)	<div>2</div>	<div>y = x</div>		
	<div>3</div>	<div>z = y</div>		
	<div>4</div>	<div>izpis z</div>	<div></div>	
	<div>#</div>	<div>Program</div>	<div>Izpis</div>	<div>★★★</div>
	<div>1</div>	<div>x = 3</div>		
	<div>2</div>	<div>y = x</div>		
č)	<div>3</div>	<div>z = x</div>		
	<div>4</div>	<div>y = z</div>		
	<div>5</div>	<div>x = 1</div>		
	<div>6</div>	<div>izpis x</div>	<div></div>	
	<div>7</div>	<div>izpis y</div>	<div></div>	

## 10.2.2 Vstavi spremenljivko

Vstavi pravilne spremenljivke.

★★★★

a)

#	Program	Izpis
1	x = 1	
2	y = 2	
3	<div></div> = x	
4	<div></div> = y	
5	<div></div> = z	
6	izpis x	2
7	izpis y	1

★★★★

b)

#	Program	Izpis
1	x = 2	
2	y = x + 1	
3	z = <div></div> + 1	
4	<div></div> = y * y	
5	z = <div></div> + y	
6	izpis x	9
7	izpis y	3
8	izpis z	12

Vstavi pravilne spremenljivke.

★★★

#	Program	Izpis
1	x = 1	
2	y = 2	
3	x = y	
4	<input type="text"/> = x	
5	izpis x	2
6	izpis y	2

a)

★★★★

#	Program	Izpis
1	y = 2	
2	z = y - 2	
3	<input type="text"/> = 3	
4	<input type="text"/> = z + x + y	
5	izpis x	5
6	izpis y	2
7	izpis z	0

b)

★★★★



#	Program	Izpis
1	x = 1	
2	y = 2	
3	<input type="text"/> = <input type="text"/> + <input type="text"/>	
4	izpis x	3
5	izpis y	2

c)





Vstavi pravilne spremenljivke.

★★★★★

#	Program	Izpis
1	x = 1	
2	y = 2	
3	z = y	
4	w = 3	
5	x = x + z	
6	w = w + y + x	
7	 = w + 1	
8	 = w + 1	
9	izpis w	10

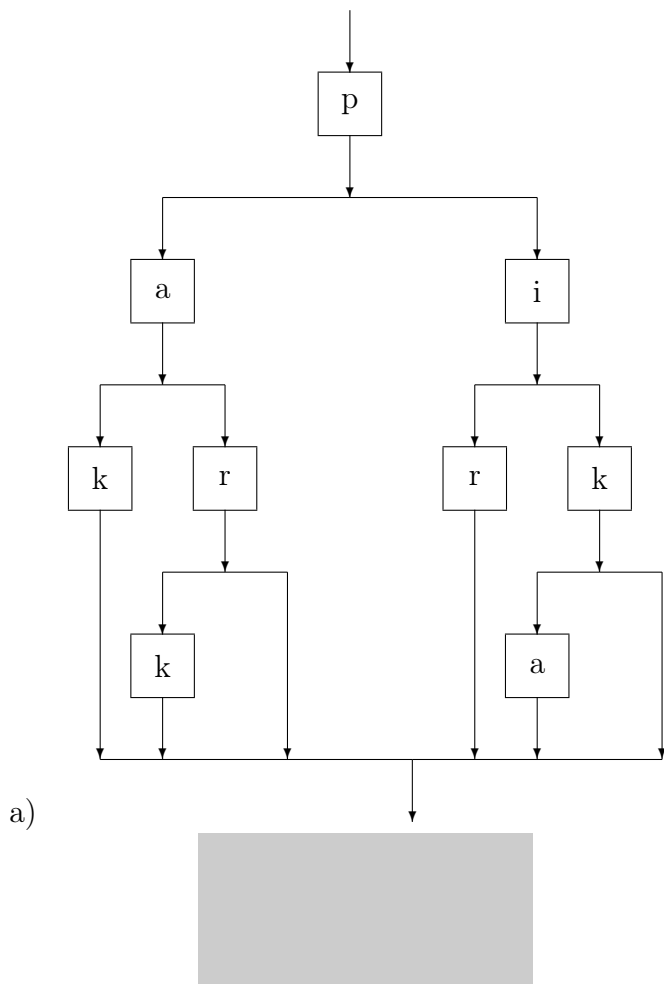
★★★★★

#	Program	Izpis
1	x = 1	
2	y = x + 1	
3	z = y + 1	
4	w = z + 1	
5	w = x + y + z + w	
6	w = w - y - x - z	
7	 = w - 2	
8	 = w + 2	
9	izpis x	4
9	izpis z	3

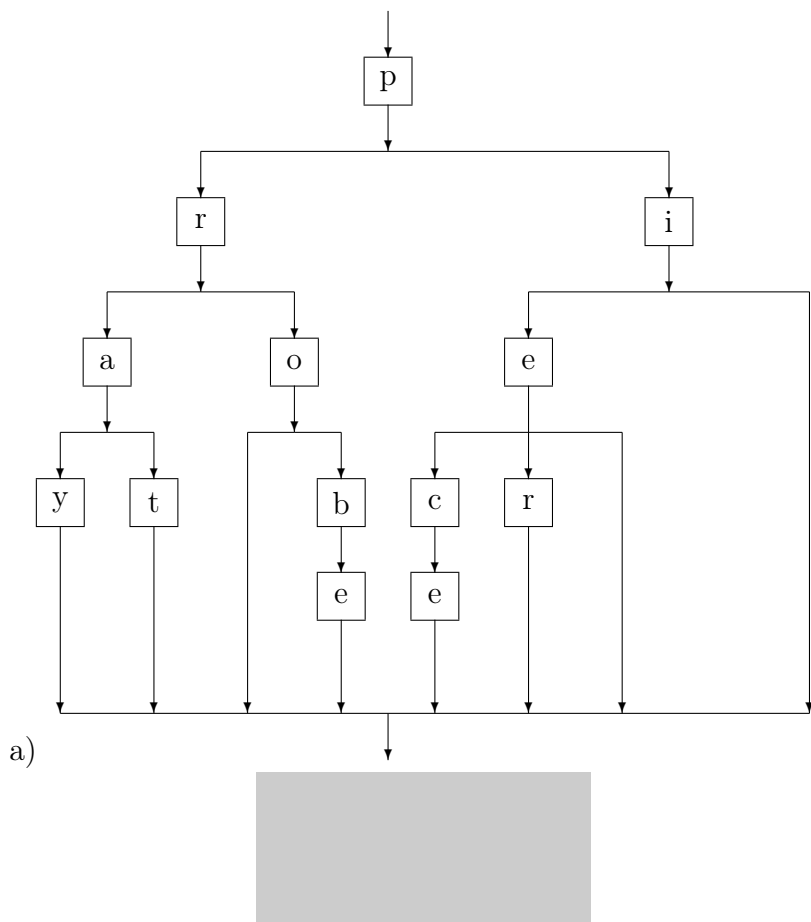
## 10.3 Vejitve

### 10.3.1 Sestavljanje besed

V sivo celico zapiši vse besede, ki jih je možno sestaviti.



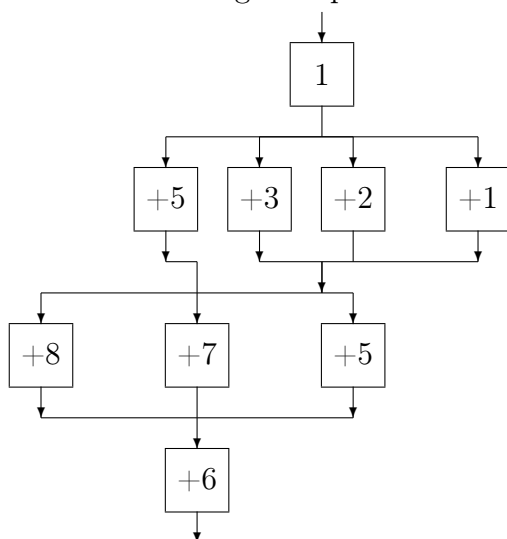
V sivo celico zapiši vse besede, ki jih je možno sestaviti.



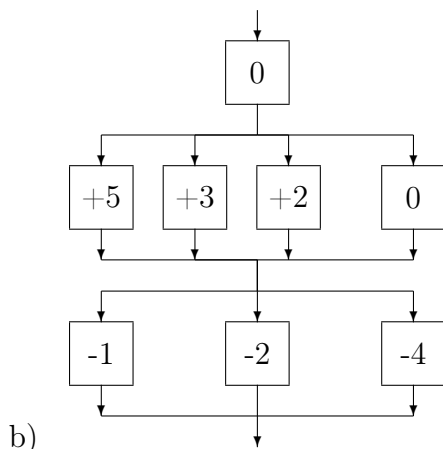
### 10.3.2 Izračunaj možne rezultate

Zapiši vse možne rezultate diagrama poteka.

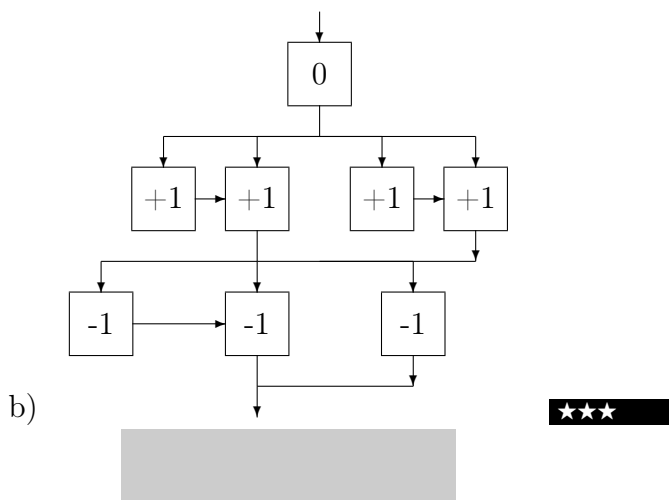
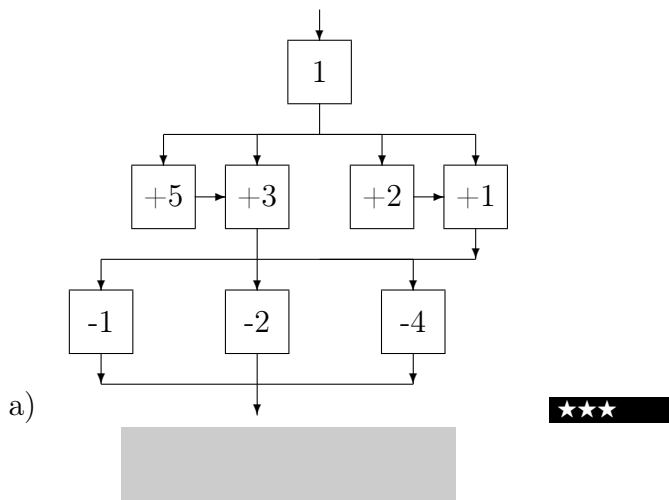
★★★



★★★

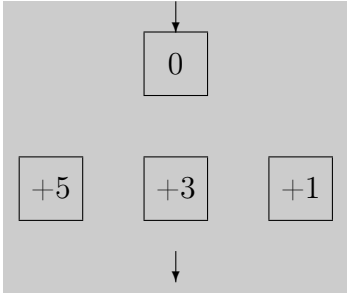




Zapiši vse možne rezultate diagrama poteka.

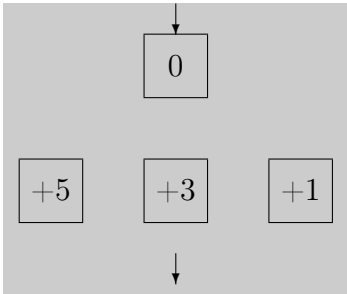



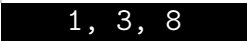
### 10.3.3 Poveži

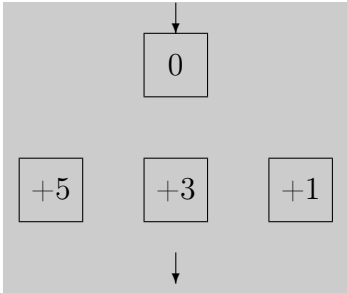

Poveži diagram tako, da so možni samo navedeni rezultati.

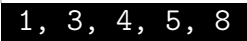
a)  



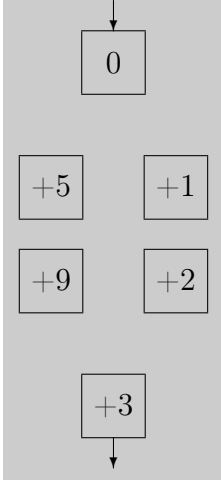

b)  



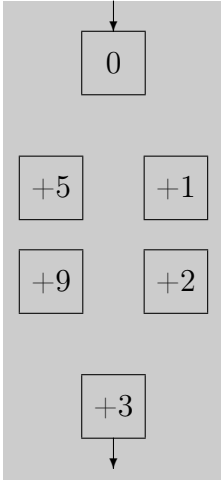

c)  



Poveži diagram tako, da so možni samo navedeni rezultati.

a)  

5, 17

b)  

5, 11, 15, 17

Poveži diagram tako, da so možni samo navedeni rezultati.

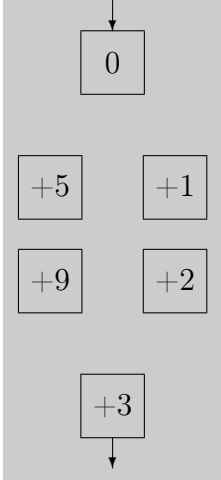


a)   

Diagram a) shows a path starting at 0, moving down to +5, then right to +1, then down to +2, then left to +3. The path is highlighted with a thick black line. The numbers are arranged in a grid: 0 at the top, +5 and +1 in the middle, +9 and +2 in the bottom, and +3 at the very bottom. Arrows indicate the path from 0 to +3.

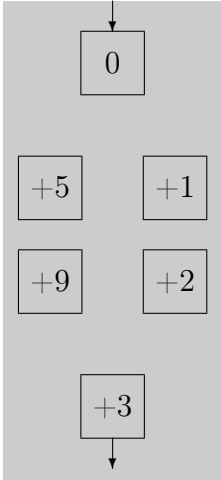

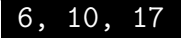
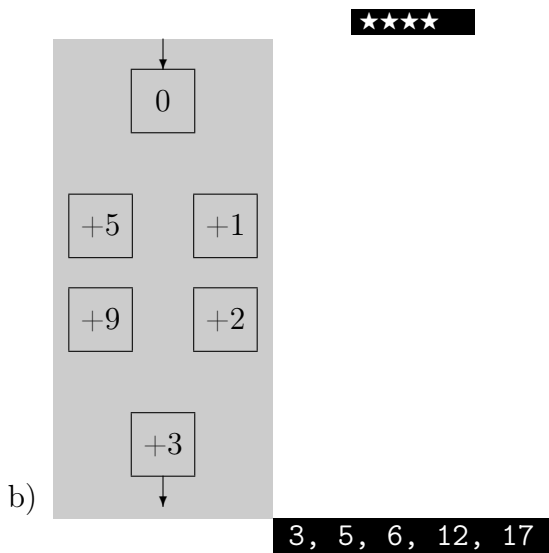
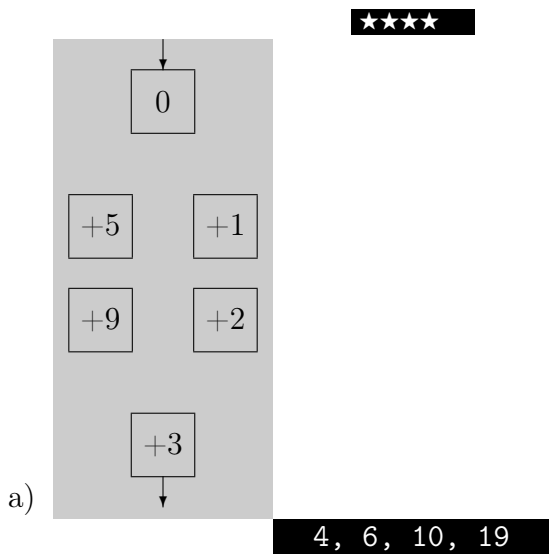
b)   

Diagram b) shows a path starting at 0, moving down to +5, then right to +1, then down to +2, then left to +3. The path is highlighted with a thick black line. The numbers are arranged in a grid: 0 at the top, +5 and +1 in the middle, +9 and +2 in the bottom, and +3 at the very bottom. Arrows indicate the path from 0 to +3.

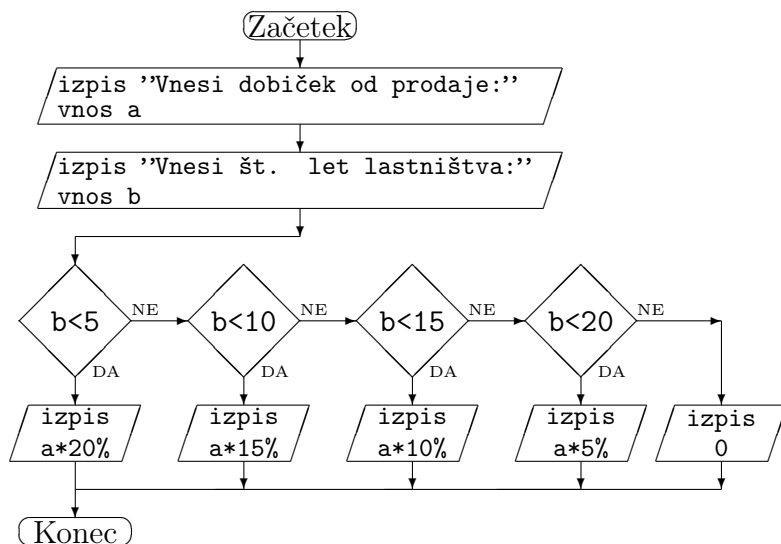


Poveži diagram tako, da so možni samo navedeni rezultati.



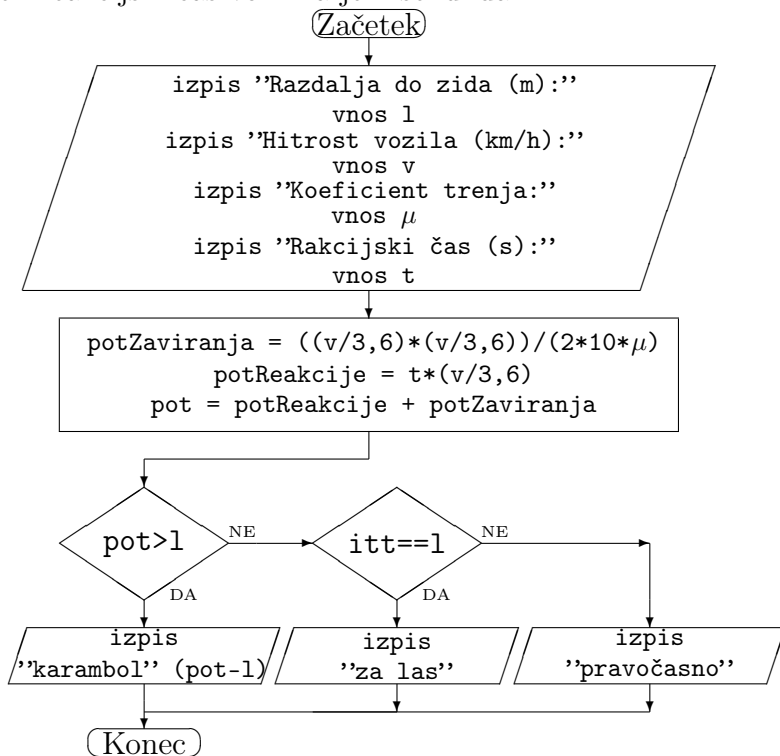
### 10.3.4 Zapiši izpis

Zapiši izpis programa (vhod je podan, kot desna stran). Program opisuje izračun davka od prodaje delnic. Uporabnik mora vnesti dobiček pri prodaji in čas lastništva delnic.



	Vnesen dobiček (a)	Vnesen čas (b)	Izpis vrednosti davka	
	100	3	20	
a)	1000	4		★★
b)	100	8		★★
c)	300	10		★★
č)	200	20		★★
d)	100	5		★★

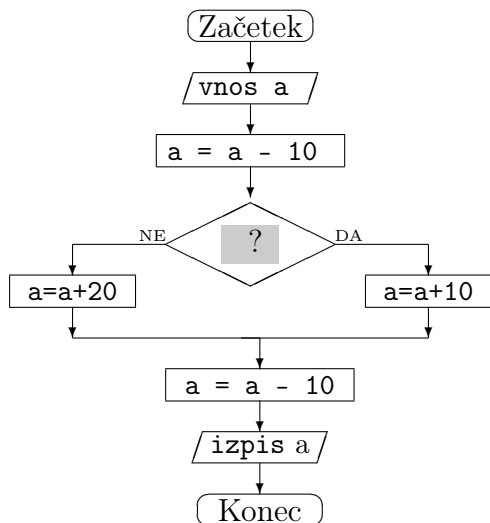
Zapiši izpis programa (vhod je podan, kot desna stran). Program opisuje izračun poti ustavljanja avtomobila, ki se vozi proti zidu. Pri izračunu potrebujemo koeficient trenja med pnevmatiko in asfaltom ( $\mu$ ). Informativni povprečni koeficienti trenja med pnevmatiko in asfaltom so: 0,7 za suh asfalt, 0,4 za moker asfalt in 0,1 za poledenel asfalt. Povprečen reakcijski čas voznika je 1 sekunda.



	l	v	$\mu$	t	Izpis	
	50	36	0,1	2	karambol 20	
a)	50	36	0,2	2		★★★
b)	100	100	0,7	1		★★★
c)	100	100	0,5	1		★★★
č)	60	72	0,5	1		★★★

### 10.3.5 Izberi pogoj

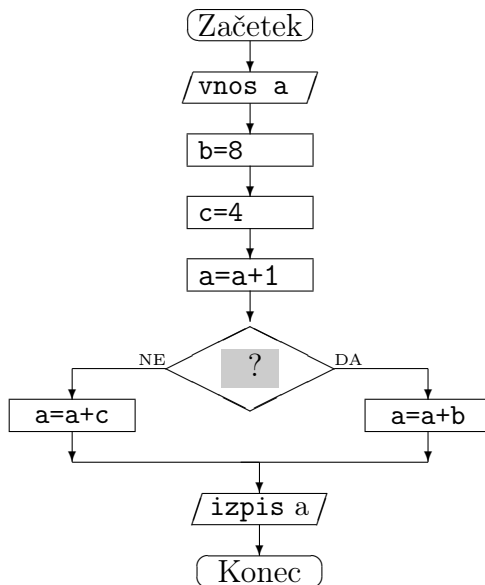
Za podan vhod izberi pogoje tako, da ustrezajo podanem izpisu. Vhod in izpis sta določena v tabeli.



Izberi med naslednjimi pogoji:  $a=0$ ,  $a<10$ ,  $a\geq 10$  in  $a>-5$ .

	Vnos a	Izpis a	Pogoj	
a)	5	-5		★★
b)	6	-4		★★
c)	5	5		★★
č)	20	10		★★
d)	-10	-20		★★★
e)	20	20		★★★

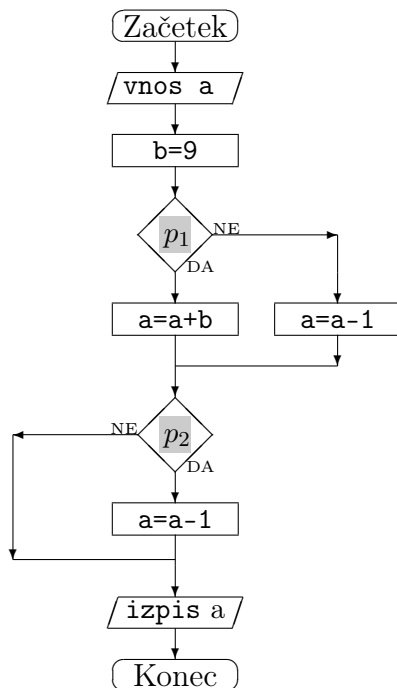
Pravilno izberi pogoj-e.



Izberi med pogoji:  $a > b$ ,  $a < b$ ,  $a < c$  in  $b == c$ .  
Pravilen odgovor-e, vpišite v sivo polje tabele.

	Vnos a	Izpis a	Pogoj	
a)	1	10		★★
b)	1	6		★★
c)	4	9		★★★
č)	5	14		★★★
d)	10	19		★★★
e)	16	21		★★★

Izberi pare pogojev vejitev ( $p_1, p_2$ ), tako da dobimo ob podanem vhodu, željen izpis.

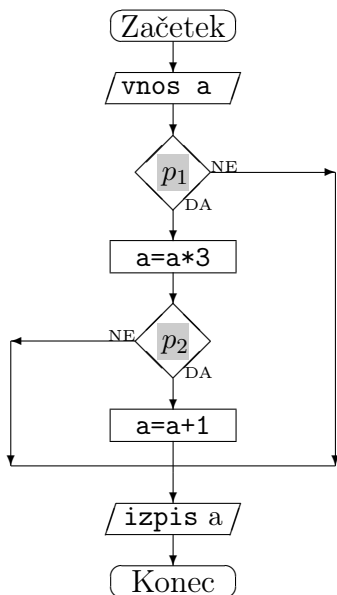


Izberi med pogoji:  $a < 4$ ,  $a > b$  in  $a == 8$ .

Vnos a Izpis a Pogoj ( $p_1, p_2$ )

	Vnos a	Izpis a	Pogoj ( $p_1, p_2$ )	
a)	8	16		★★
b)	1	9		★★★
c)	1	10		★★★
č)	10	9		★★★
d)	10	8		★★★

Izberi pare pogojev vejitev ( $p_1, p_2$ ), tako da dobimo ob podanem vhodu, željen izpis.

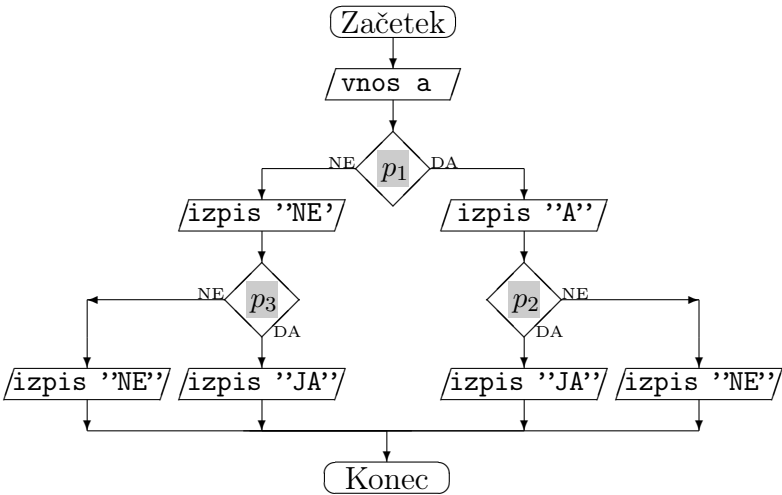


Izberi med pogoji:

- $a > 3$ ,
- $a == a$  in
- $a < 3$ .

Vnos a Izpis a Pogoj ( $p_1, p_2$ )

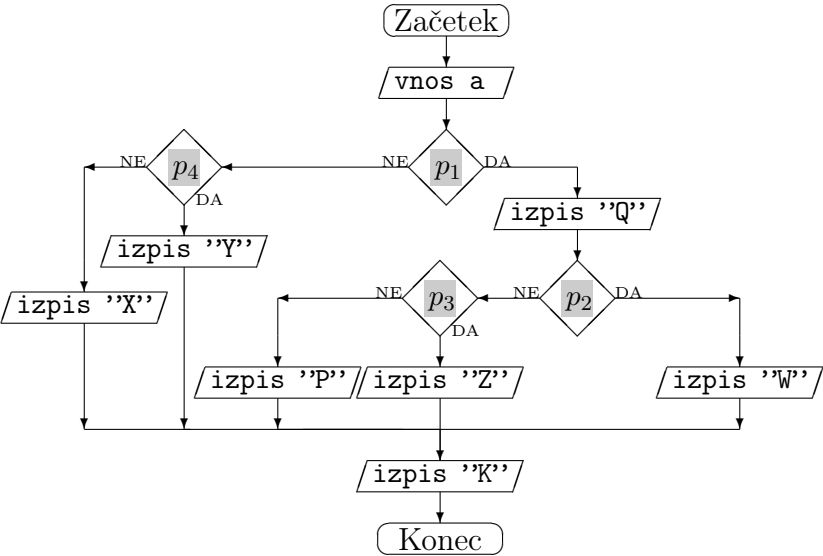
	Vnos a	Izpis a	Pogoj ( $p_1, p_2$ )
a)	1	4	★★★★
b)	3	10	★★★★
c)	3	3	★★★★
č)	5	15	★★★★
d)	3	9	★★★★



V tabeli povežite oznake vejitev diagrama, s pripadajočim pogojem (desna stran), tako da bo program ob podanem vhodu (leva tabela), izpisal željen izpis. Vsak pogoj lahko uporabite samo enkrat.

		★★★		
	Vnos a	Izpis a	Oznaka vejitve	Pogoj
a)	7	ANE	$p_1$	$a < 3$
	10	AJA	$p_2$	$a > 6$
			$p_3$	$a > 8$
		★★★		
	Vnos a	Izpis a	Oznaka vejitve	Pogoj
b)	2	NEJA	$p_1$	$a < 3$
	4	NENE	$p_2$	$a > 6$
			$p_3$	$a > 8$
		★★★		
	Vnos a	Izpis a	Oznaka vejitve	Pogoj
c)	2	ANE	$p_1$	$a < 3$
	8	NEJA	$p_2$	$a > 6$
			$p_3$	$a > 8$





V tabeli povežite oznake vejitev diagrama, s pripadajočim pogojem (vsak pogoj lahko izberete samo enkrat).

★★★			
a)	Vnos a	Izpis a	Oznaka vejitve
	5	QWK	p1
	8	QZK	p2
	12	QPK	p3
			p4
			Pogoj
			a > 3
			a == 5
			a < 14
			a < 5

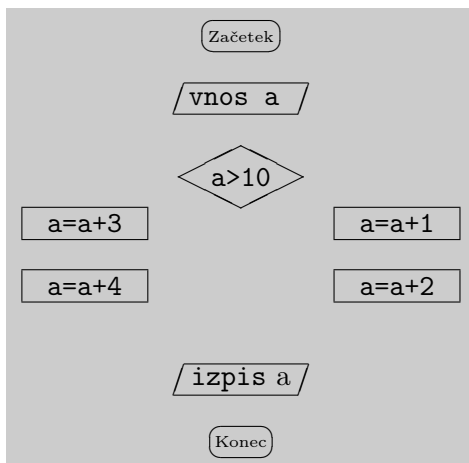
★★★			
b)	Vnos a	Izpis a	Oznaka vejitve
	5	YK	p1
	4	QWK	p2
	3	QPK	p3
			p4
			Pogoj
			a > 3
			a == 5
			a < 14
			a < 5

### 10.3.6 Poveži

S puščicami poveži diagram tako, da ustreza pogojem iz tabele (črne celice).

★★★

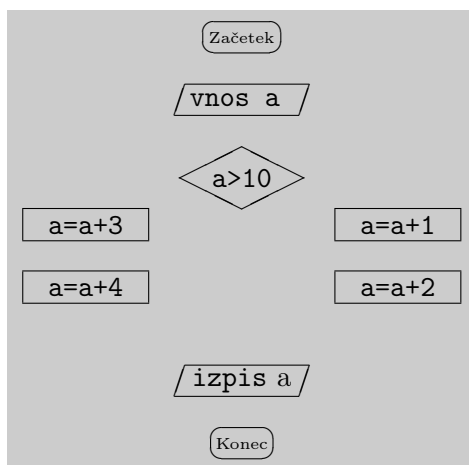
	Vnos	Izpis
a)	7	14
	11	14



Pri povezovanju ni nujno povezati vse simbole iz slike.

★★★★

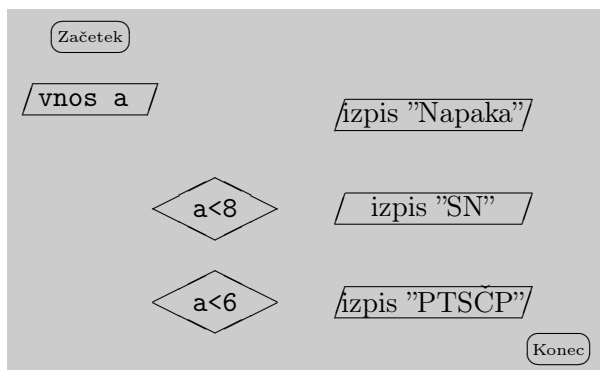
	Vnos	Izpis
b)	7	8
	11	14



Poveži diagram tako, da bo program v primeru vnosa števila od 1 do 5 izpisal prve črke izbranega dneva v tednu, tj. od ponedeljka do petka (PTSČP). Pri vnosu števil 6 ali 7 izpiše prvi črki sobote in nedelje (SN). Za števila večja od 7 izpiše napaka.

★★★★

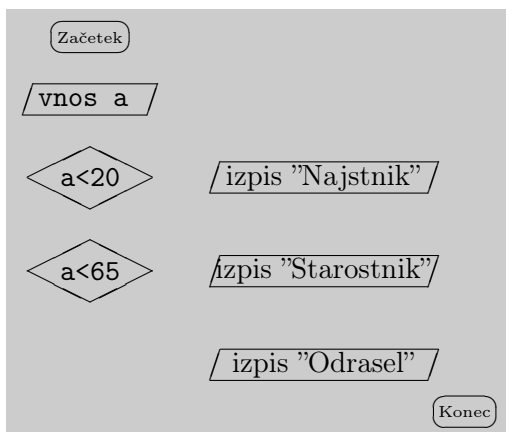
Vnos	Izpis
2	PTSČP
6	SN
8	Napaka



Poveži diagram tako, da bo glede na vneseno starost izpisal: "Mladostnik", "Odrasel" ali "Starostnik".

★★★★

Vnos	Izpis
16	Mladostnik
67	Starostnik
30	Odrasel

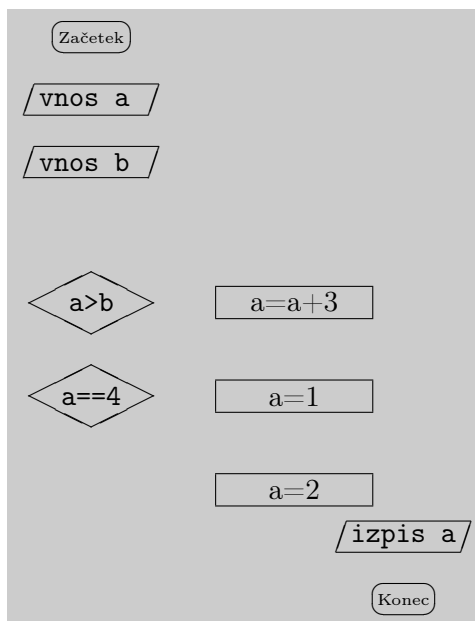


Poveži diagram, tako da ustreza pogojem v tabeli. Program prebere dve spremenljivki (a in b), ter na koncu izpiše spremenljivko a.



Vnos		Izpis
a	b	a
2	3	1
3	3	2
2	6	7

a)

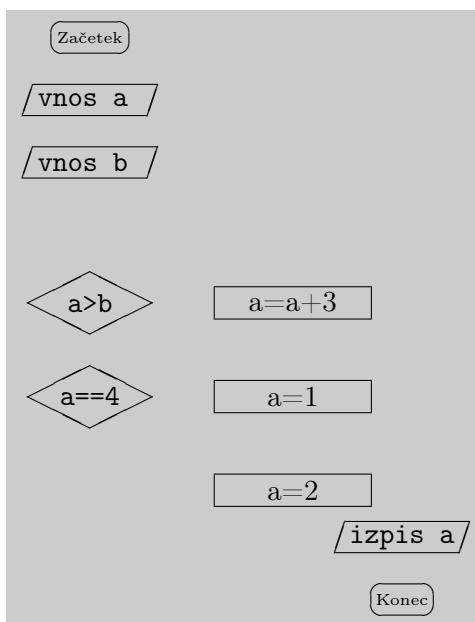


Poveži diagram, tako da ustreza pogojem v tabeli. Program prebere dve spremenljivki (a in b), ter na koncu izpiše spremenljivko a. Bodite pozorni na izpis.

★★★★

Vnos		Izpis
a	b	a
4	3	2
5	3	1
3	2	6

a)

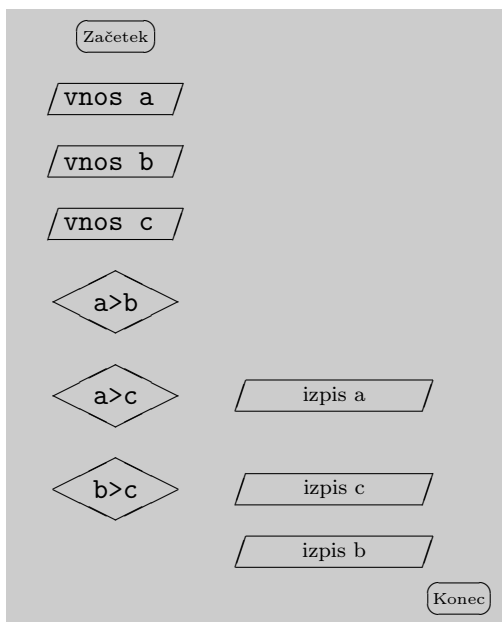


Poveži diagram tako, da bo omogočal vnos števil a, b in c. Izpiše naj največje število izmed vnesenih.



Vnos			Izpis
a	b	c	
2	4	3	4
1	1	3	3
5	4	3	5
1	0	0	1

a)

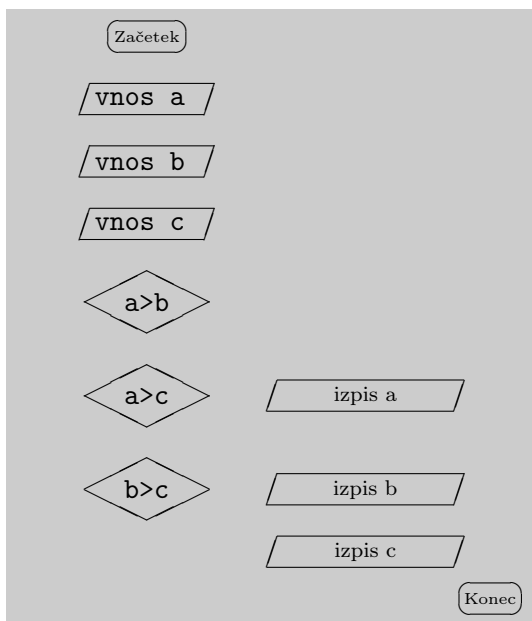


Poveži diagram tako, da bo omogočal vnos števil a, b in c. Izpiše naj najmanjše število izmed vnesenih.



Vnos			Izpis
a	b	c	
2	4	3	2
1	1	3	1
5	4	3	3
1	0	0	0

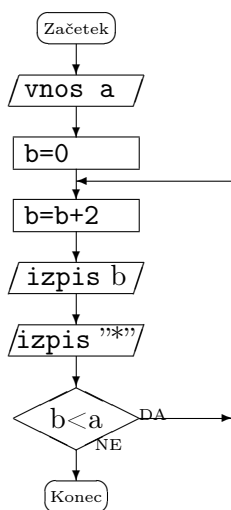
a)



## 10.4 Ponavljanje

### 10.4.1 Zapiši izpis

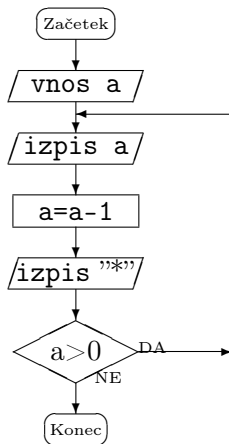
V sive celice zapiši izpis programa (vhod je podan v tabeli). Izpis bo potekal brez presledkov in lomljenja vrstic.



	Vnos a	Izpis	
	3	2*	
a)	2		***
b)	3		***
c)	5		***
č)	10		***
d)	13		***

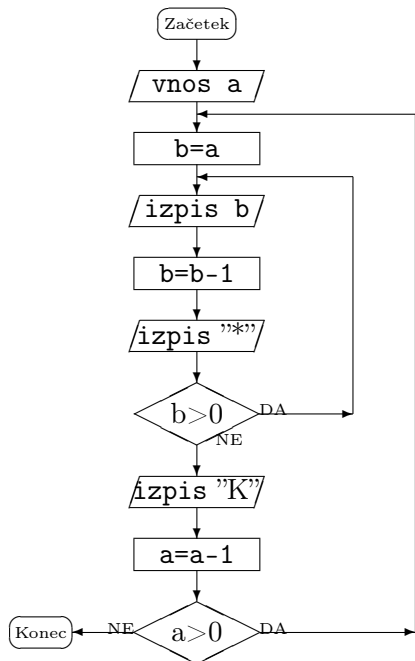


V sive celice zapiši izpis programa (vhod je podan v tabeli). Izpis bo potekal brez presledkov in lomljenja vrstic.



	Vnos a	Izpis	
	0	0*	
a)	2		***
b)	6		***
c)	3		***
č)	-1		***
d)	-4		***

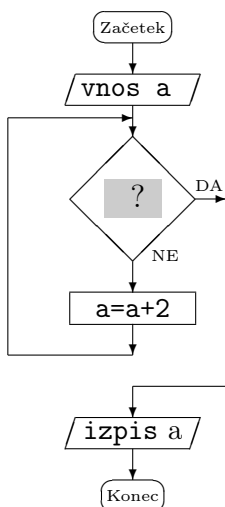
V sive celice zapiši izpis program (vhod je podan v tabeli). Izpis bo potekal brez presledkov in lomljenja vrstic.



	Vnos a	Izpis	
	0	0*K	
a)	2		★★★
b)	3		★★★
c)	-1		★★★
č)	4		★★★
d)	1		★★★

## 10.4.2 Izberi pogoj

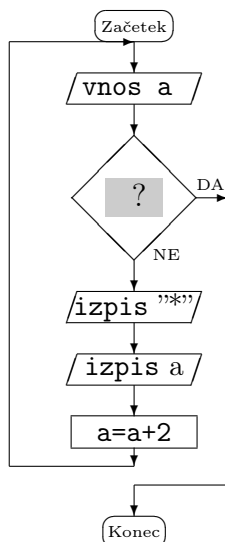
Izberi pogoj-e diagrama tako, da dobimo željen izpis.



Izberi med pogoji:  $a < 3$ ,  $a > 3$ ,  $a > 8$ ,  $a == 8$  in  $a < > 8$ .

	Vnos a	Izpis a	Pogoj	
a)	2	8		★★
b)	2	10		★★
c)	8	10		★★
č)	1	5		★★
d)	2	4		★★
e)	8	8		★★

Izberi pogoj-e diagrama tako, da dobimo željen izpis. Kot rezultat bomo izpisovali zvezdice in vrednost spremenljivke a. Pri izpisu bomo vse izpisovali brez presledkov in v isto vrstico.

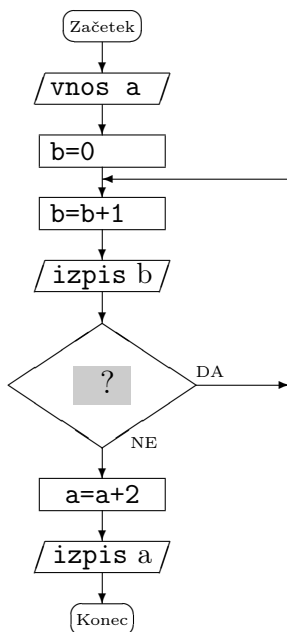


Izberi med pogoji:  $a < 3$ ,  $a > 3$ ,  $a > 8$ ,  $a == 8$  in  $a < > 8$ .

Vnos a Izhis (zvezdice) Pogoj

	Vnos a	Izhis (zvezdice)	Pogoj
a)	2	*2	***
b)	1	*1*3	***
c)	6	*6	***
č)	6	*6*8	***
d)	0	*0*2*4*6*8	***
e)	4	*4*6	***

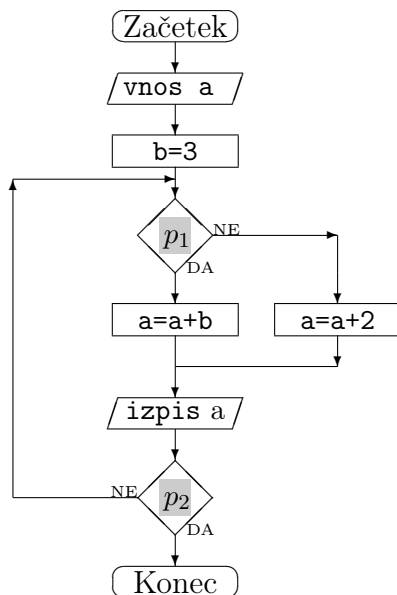
Izberi pogoj-e diagrama tako, da dobimo željen izpis. Bodite pozorni, da imamo opravka s spremenljivkama a in b. Izpis poteka brez presledkov in lomljenja vrstice.



Izbiramo med pogoji:  $b > 3$ ,  $b < 4$ ,  $b == 1$ ,  $a > b$  in  $a <> b$ .

	Vnos a	Izpis	Pogoj	
a)	0	12342		★★★★★
b)	3	1235		★★★★★
c)	5	123457		★★★★★
č)	4	12346		★★★★★
d)	2	12344		★★★★★
e)	1	123		★★★★★

Izberi pare pogojev ( $p_1, p_2$ ) diagrama tako, da dobimo ob podanem vходу, željen izpis. Izpis vrednosti spremenljivke  $b$  je brez presledkov.



Izbiri med pogoji:  $a > 3$ ,  $a > 7$  in  $b == 3$ .

	Vnos a	Izpis	Pogoj	
a)	0	247		★★★★★
b)	-1	14		★★★★★
c)	5	8		★★★★★
č)	1	47		★★★★★
d)	6	8		★★★★★

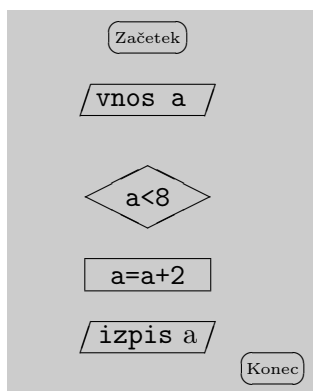
### 10.4.3 Poveži

S puščicami poveži diagram tako, da bo program ob podanih vseh izpisal željen izpis. Pri povezovanju uporabite tudi zanke (povratne puščice).

★★★

a)

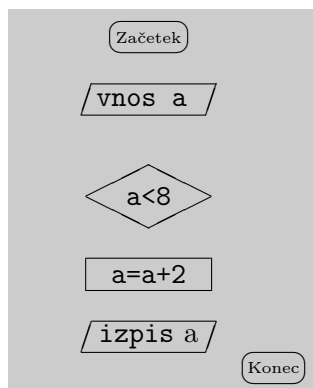
Vnos	Izpis
1	3579
2	468
7	9
8	brez



★★★

b)

Vnos	Izpis
1	1357
2	246
7	7
8	8

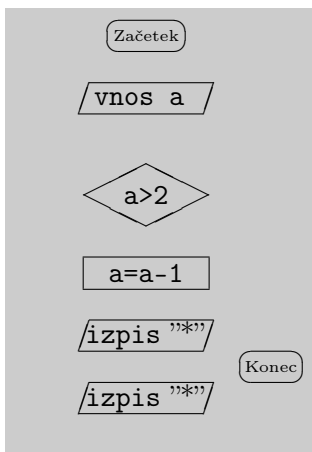


S puščicami poveži diagram tako, da bo program ob podanih vhodih izpisal željen izpis. Pri povezovanju uporabite tudi zanke (povratne puščice). Ni potrebno povezati vse simbole diagrama poteka.

★★★

a)

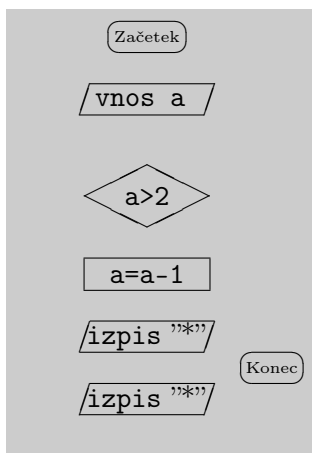
Vnos	Izpis
3	**
5	*****
1	**
0	**



★★★

b)

Vnos	Izpis
3	*
5	***
6	*****
0	*



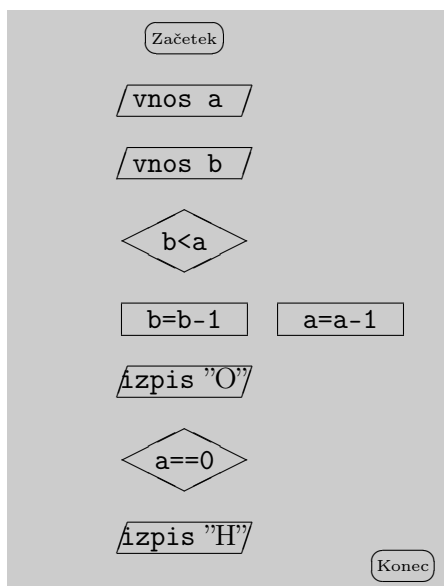


Poveži simbole diagrama.

★★★

a)

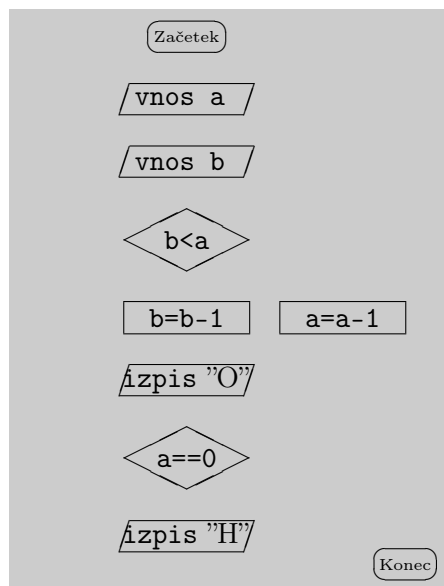
Vnos		Izpis
a	b	
1	0	HHO
1	2	HHOHOHO
2	0	H
0	2	HOHOHO



★★★★

b)

Vnos		Izpis
a	b	
1	0	OH
1	2	OOOH
2	0	OOH
0	2	OH

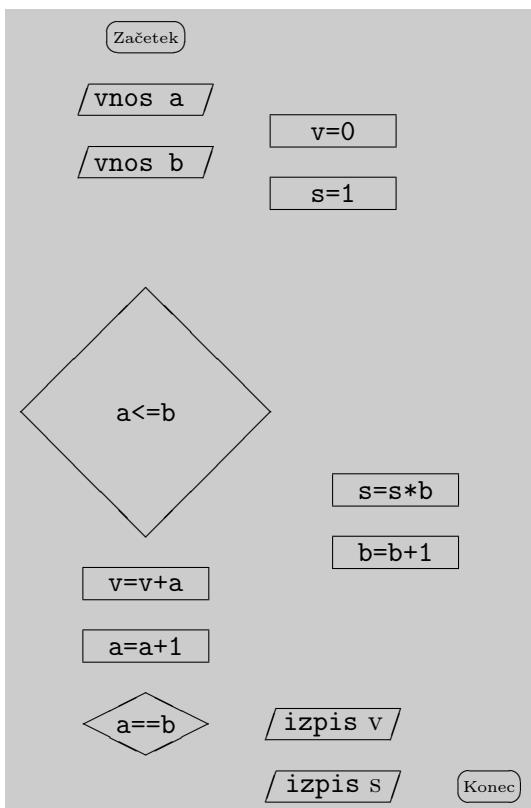


Poveži diagram tako, da ustreza pogojem v spodnji tabeli. Program prebere dve spremenljivki (a in b), ter ju na koncu izpiše. Uporabite zanke.



Vnos		Izpis
a	b	
1	3	3 1
3	1	0 2
0	3	3 1
3	0	0 0
4	1	0 6

a)



Konec priloge.

*progo.crepinsek@gmail.com*

---

# 11.

*Če se ne držiš pravil igre,  
zakaj želiš sodelovati v njej?*  
*- Latinski pregovor -*

## Rešitve

Rešitve različnih nalog so podane s pomočjo tabele, ki opiše stran naloge, oznako naloge in njeno rešitev. Vse rešitve so podane od zgoraj navzdolj, to pomeni, da v primeru, ko je za rešitev potrebnih več vrstic so lete ločene s podpičjem in prva vrednost pomeni prvo vrstico.

Tabela 11.1: Rešitve nalog

Stran	Rešitev
25	a x y y x x
25	b x y y z
25	c w x w z
25	č w z y w
25	d w w y y w
25	e x z w z z
27	a birabara biri ba
27	b biribari bara ri

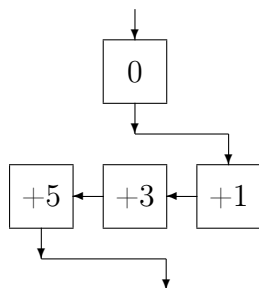
Tabela 11.1 – Se nadaljuje

Stran	Rešitev
27	c birabari bira bi
28	a ajjavaja vaja ja
28	b vaavajja java aj
28	c vajajava jaa av
29	a cacaba ba
29	b babccaba cab ba
29	c bacabacabc caba bc
32	a $x = "cd"$
32	b $x = "aa"$ in $y = "bb"$
32	c $x = "ab"$ in $y = "ba"$
32	č $x = "a"$ in $y = "bbaa"$
35	a $x = "ab"$ , $y = "ba"$ in $z = "ac"$
35	b $x = "dd"$ , $y = "ad"$ in $z = "aa"$
35	a $x = "aaa"$ , $y = "aa"$ in $z = "ab"$
35	b $x = "00"$ , $y = "01"$ in $z = "10"$
35	a $x = "110"$ , $y = "1"$ , $z = "10"$ in $w = "00"$
35	b $x = "a"$ , $y = "b"$ , $z = "aa"$ in $w = "ba"$



Tabela 11.1 – Se nadaljuje

Stran Rešitev



- 63 c
- 71 a Ostri kot
- 71 b Vdrti kot
- 71 c Ostri kot
- 71 č Topi kot
- 71 d Topi kot Rešitev ni optimalna. Moralo bi izpisati pravi kot, za kar bi morali dodati nalogi dodaten pogoj (če  $a=90$ ) in izpis.
- 72 a  $100/(2 * 2) = 25$ ; izpiše: idealna teža
- 72 b  $60/(1,7 * 1,7) = 20,76$ ; izpiše: idealna teža
- 72 c  $90/(1,82 * 1,82) = 27,17$ ; izpiše: prekomerna teža
- 72 č  $50/(1,67 * 1,67) = 17,93$ ; izpiše: nezadostna teža
- 72 d vnesite svoje podatke za težo in višino; izpiše:
- 
- 73 a  $a > 100$
- 73 b  $a > 130$
- 73 c  $a < 90$
- 73 č  $a < 90$
- 74 a  $a > 5$  ali  $a > 6$  ali  $a > 7$
- 74 b  $a > 3$  ali  $a > 4$  ali  $a > 5$
- 74 c  $a > 3$  ali  $a > 4$  ali  $a > 5$  ali  $a > 6$  ali  $a > 7$
- 74 č noben pogoj ne ustreza
- 75 a  $a == b$  ali  $a < b$



Tabela 11.1 – Se nadaljuje

Stran	Rešitev		
75	b $a==b$		
75	c $a>b$ ali $a>10$		
75	č $a>b$ ali $a>10$		
75	d $a>b$ ali $a>10$		
75	e noben pogoj ne ustreza		
76	a $(a==8, a<4)$ ali $(a>7, a<4)$		
76	b $(a<4, a<4)$		
76	c $(a==8, a<4)$ ali $(a==8, a==8)$ ali $(a<4, a==8)$ ali $(a<4, a<4)$		
76	č $(a<4, a>7)$ ali $(a<4, a==8)$		
77	a $(a==2, a>3)$		
77	b $(a>3, a==2)$		
77	c $(a>3, a==12)$		
77	č $(a==12, a==2)$		
77	d $(a==12, a>3)$		
77	e poljuben pogoj		
<table> <tr> <th>Oznaka vejitve</th><th>Pogoj</th></tr> </table>		Oznaka vejitve	Pogoj
Oznaka vejitve	Pogoj		
78	a		
<table> <tr> <th>Oznaka vejitve</th><th>Pogoj</th></tr> </table>		Oznaka vejitve	Pogoj
Oznaka vejitve	Pogoj		
78	b		
<table> <tr> <th>Oznaka vejitve</th><th>Pogoj</th></tr> </table>		Oznaka vejitve	Pogoj
Oznaka vejitve	Pogoj		
79	a		

Tabela 11.1 – Se nadaljuje

Stran Rešitev

		Oznaka vejitve	Pogoj
79	b	$p_1 \longrightarrow$	$a > 5$
		$p_2 \longrightarrow$	$a < 12$
		$p_3 \longrightarrow$	$a == 14$
		Oznaka vejitve	Pogoj
79	c	$p_1 \searrow$	$a > 5$
		$p_2 \swarrow$	$a < 12$
		$p_3 \swarrow$	$a == 14$

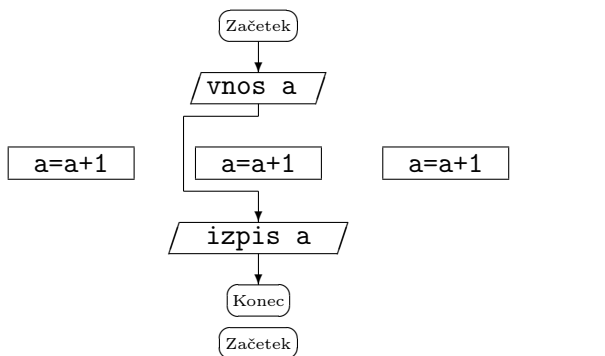
```
graph TD; Start([Začetek]) --> Input[/vnos a/]; Input --> Loop1[a=a+1]; Loop1 --> Loop2[a=a+1]; Loop2 --> Loop3[a=a+1]; Loop3 --> Output[/izpis a/]; Output --> End([Konec]);
```

83	a	<pre>graph TD; Start([Začetek]) --&gt; Input[/vnos a/]; Input --&gt; Loop1[a=a+1]; Loop1 --&gt; Loop2[a=a+1]; Loop2 --&gt; Output[/izpis a/]; Output --&gt; End([Konec]); Standalone[a=a+1];</pre>
83	b	<pre>graph TD; Start([Začetek]) --&gt; Input[/vnos a/]; Input --&gt; Loop1[a=a+1]; Loop1 --&gt; Loop2[a=a+1]; Loop2 --&gt; Output[/izpis a/]; Output --&gt; End([Konec]);</pre>

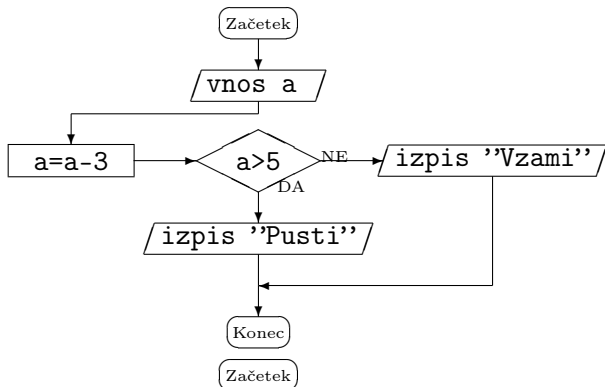
Tabela 11.1 – Se nadaljuje

Stran Rešitev

83 c



84 a



84 b

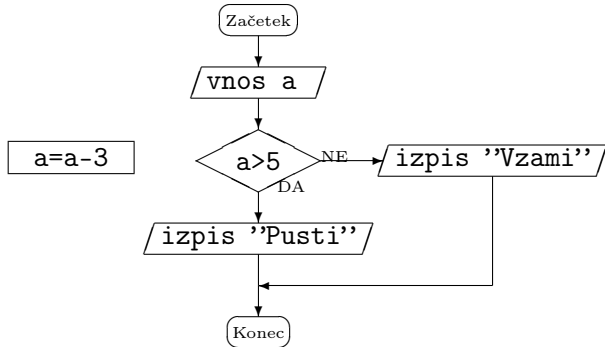
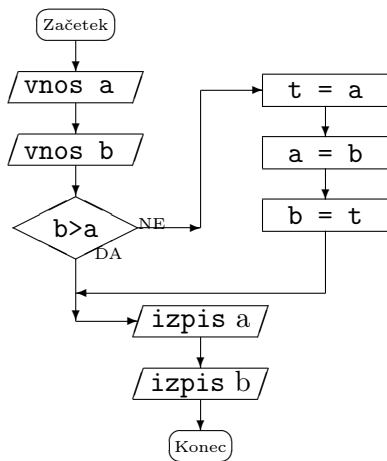
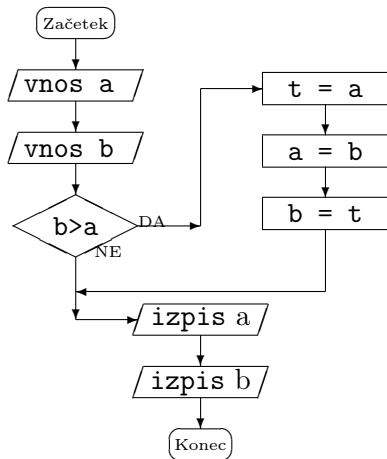


Tabela 11.1 – Se nadaljuje

Stran Rešitev



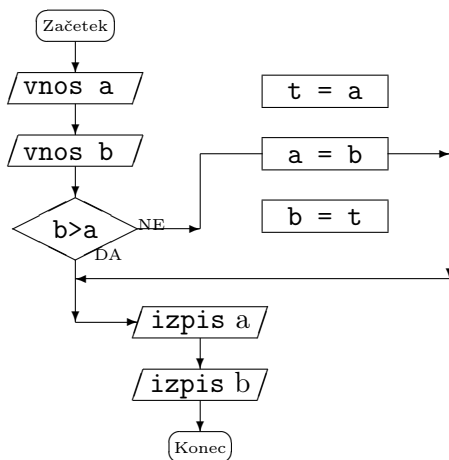
85 a



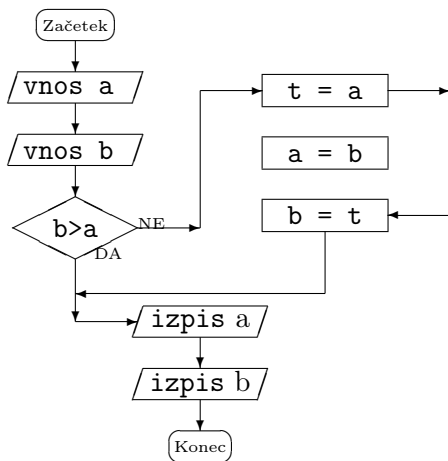
85 b

Tabela 11.1 – Se nadaljuje

Stran Rešitev



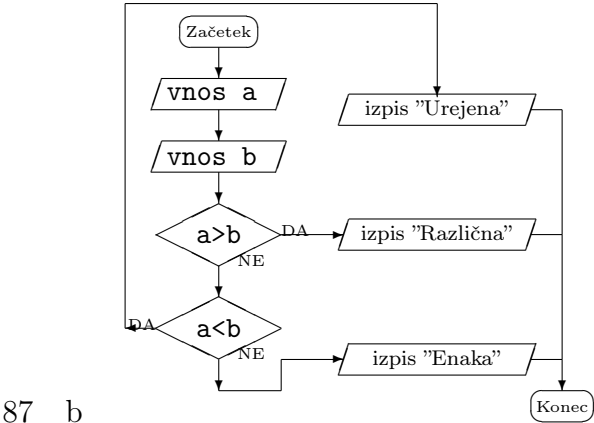
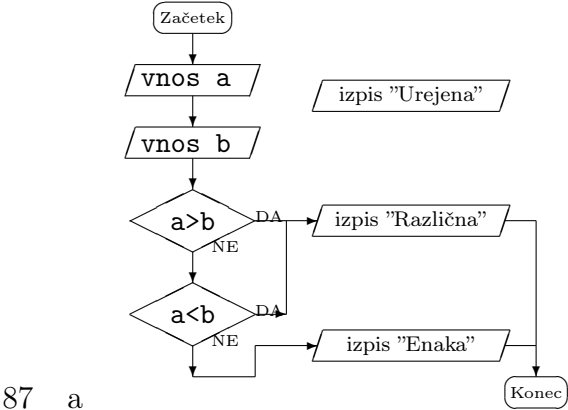
86 a



86 b

Tabela 11.1 – Se nadaljuje

Stran	Rešitev
-------	---------



100 a 1, 0, 0, -1

Tabela 11.1 – Se nadaljuje

Stran		Rešitev																																																								
		<table><tr><th>Korak</th><th><i>Ukaz</i></th><th>a</th><th>Izpis</th></tr><tr><td>1</td><td>vnos a</td><td>2</td><td></td></tr><tr><td>2</td><td>izpis a</td><td></td><td>2</td></tr><tr><td>3</td><td>a = a - 1</td><td>1</td><td></td></tr><tr><td>4</td><td>izpis a</td><td></td><td>1</td></tr><tr><td>5</td><td>a = a - 1</td><td>0</td><td></td></tr><tr><td>6</td><td>izpis a</td><td></td><td>0</td></tr><tr><td>7</td><td>a = a - 1</td><td>-1</td><td></td></tr><tr><td></td><td><i>konec</i></td><td></td><td></td></tr></table>	Korak	<i>Ukaz</i>	a	Izpis	1	vnos a	2		2	izpis a		2	3	a = a - 1	1		4	izpis a		1	5	a = a - 1	0		6	izpis a		0	7	a = a - 1	-1			<i>konec</i>																						
Korak	<i>Ukaz</i>	a	Izpis																																																							
1	vnos a	2																																																								
2	izpis a		2																																																							
3	a = a - 1	1																																																								
4	izpis a		1																																																							
5	a = a - 1	0																																																								
6	izpis a		0																																																							
7	a = a - 1	-1																																																								
	<i>konec</i>																																																									
100	b																																																									
101	a	6, 6, 6																																																								
		<table><tr><th>Korak</th><th><i>Ukaz</i></th><th>vs</th><th>a</th><th>Izpis</th></tr><tr><td>1</td><td>vs = 0</td><td>0</td><td></td><td></td></tr><tr><td>2</td><td>vnos a</td><td></td><td>2</td><td></td></tr><tr><td>3</td><td>vs = vs + a</td><td>2</td><td></td><td></td></tr><tr><td>4</td><td>a = a - 1</td><td></td><td>1</td><td></td></tr><tr><td>5</td><td>vs = vs + a</td><td>3</td><td></td><td></td></tr><tr><td>6</td><td>a = a - 1</td><td></td><td>0</td><td></td></tr><tr><td>7</td><td>vs = vs + a</td><td>3</td><td></td><td></td></tr><tr><td>8</td><td>a = a - 1</td><td></td><td>-1</td><td></td></tr><tr><td>9</td><td>izpis vs</td><td></td><td></td><td>3</td></tr><tr><td></td><td><i>konec</i></td><td></td><td></td><td></td></tr></table>	Korak	<i>Ukaz</i>	vs	a	Izpis	1	vs = 0	0			2	vnos a		2		3	vs = vs + a	2			4	a = a - 1		1		5	vs = vs + a	3			6	a = a - 1		0		7	vs = vs + a	3			8	a = a - 1		-1		9	izpis vs			3		<i>konec</i>				
Korak	<i>Ukaz</i>	vs	a	Izpis																																																						
1	vs = 0	0																																																								
2	vnos a		2																																																							
3	vs = vs + a	2																																																								
4	a = a - 1		1																																																							
5	vs = vs + a	3																																																								
6	a = a - 1		0																																																							
7	vs = vs + a	3																																																								
8	a = a - 1		-1																																																							
9	izpis vs			3																																																						
	<i>konec</i>																																																									
101	b																																																									
102	a	2, 2, 3, 6, 4, 24, 5, 24																																																								

Tabela 11.1 – Se nadaljuje

Stran		Rešitev					
		#	<i>Ukaz</i>	f	k	n	Izpis
102	b	1	f = 1	1			
		2	k = 1		1		
		3	vnos n			3	
		4	f = f * k	1			
		5	k = k + 1		2		
		6	f = f * k	2			
		7	k = k + 1		3		
		8	f = f * k	6			
		9	k = k + 1		4		
		10	izpis f				6
			<i>konec</i>				
104	a	12345					
104	b	1					
104	c	1					
105	a	12					
105	b	1					
105	c	0123456					
105	č	00123456					
105	d	000123456					
106	a	zz					
106	b	zzzz					
106	c	zzzzy					
106	č	zzzzxxy					
106	d	zzzzxy					
107	a	a==8					
107	b	a<>8 ali a<8					
107	c	a<3					
108	a	a<3					
108	b	a<3					



Tabela 11.1 – Se nadaljuje

Stran	Rešitev
108	c a<8 ali a<>8
108	č a<8 ali a<>8
108	d a==8
108	e a<8 ali a<>8
109	a b>3 ali a>6
109	b b<4
109	c b==1 ali a>6
110	a b==1 ali b<4 ali a<>b
110	b b>3
110	c b==1 ali b<4 ali a<>b
110	č b==1 ali a>6 ali b<4 ali a<>b
110	d b>3

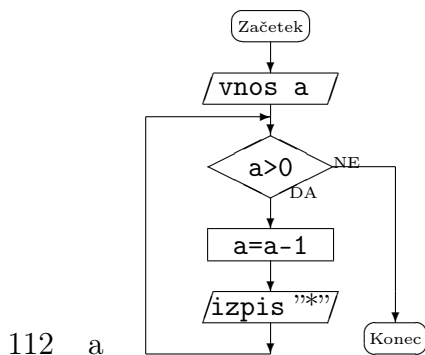


Tabela 11.1 – Se nadaljuje

Stran Rešitev

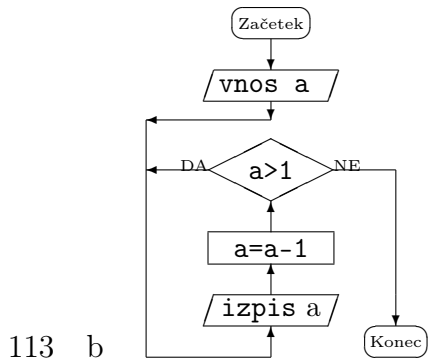
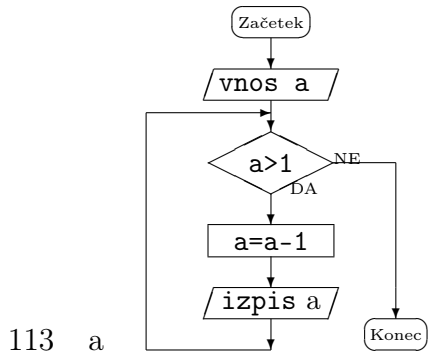
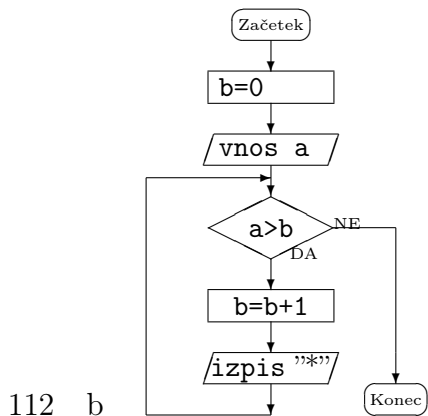
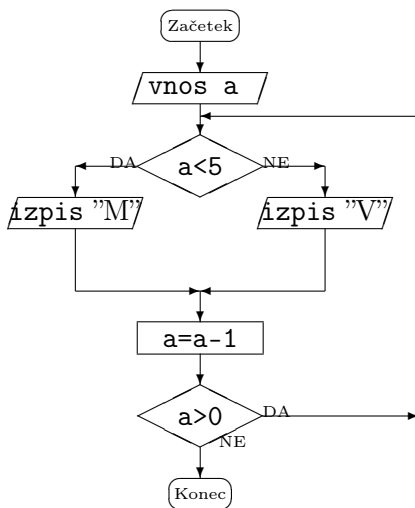


Tabela 11.1 – Se nadaljuje

Stran Rešitev

114 a



114 b

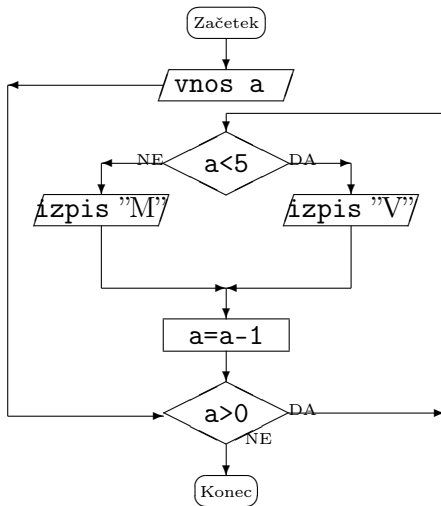
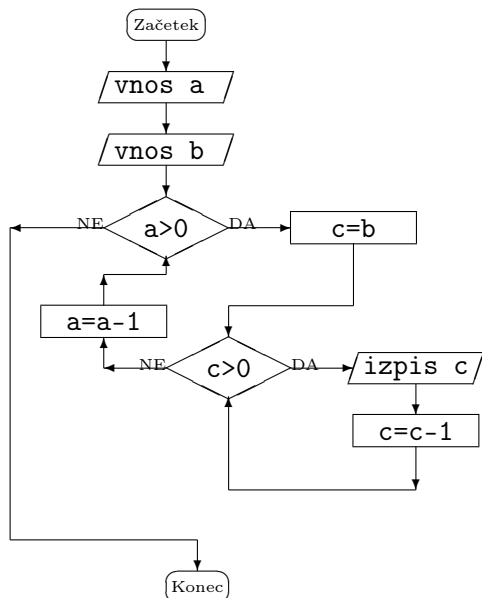


Tabela 11.1 – Se nadaljuje

Stran Rešitev



115 a  
 132 a w x x x  
 132 b i w w x  
 132 c z y k x  
 132 č z y z y  
 133 a i z i k i  
 133 b k y w z k z  
 133 c k k y y y w  
 133 č k z x x i z z  
 134 a bacabacabc  
 caba  
 134 b cacabacaba  
 babc  
 ca

Tabela 11.1 – Se nadaljuje

Stran	Rešitev
134	c cabcbabccabc bacaba ca
135	a banaara na
135	b banaara na
135	c narabanara bana
136	a rasaaar da
136	b rasada da
136	c raarsaar rasa
136	č radasada rasa
138	a octotaoc octo oc
138	b tatataocto taoc to
138	c tatataocto taoc to
139	a atama ta
139	b tamata ma

Tabela 11.1 – Se nadaljuje

Stran	Rešitev
139	c maatama ata
139	č atatatamata tama
141	a kakka ra
141	b rakara ar
141	c rakaarraka raka
141	č kakrakaarka raka
143	a pppipiiii ipii pi
143	b iiiipppppp iiii ii
143	c pipipipppppp pppii ii
144	a oppoocco cooc co
144	b opopocooc poco oc
144	c poocoococopo opopo po

Tabela 11.1 – Se nadaljuje

Stran	Rešitev
145	a babcbccabababcba babcbc ba
146	a x = "cdc"
146	b x = "cd" in y = "dc"
146	c x = "k" in y = "kak"
146	č x = "aa" in y = "ba"
147	a x = "01" , y = "10" in z = "23"
147	b x = "dd" , y = "ad" in z = "aa"
148	a x = "111" , y = "11" in z = "0"
148	b x = "01" , y = "000" in z = "10"
149	a x = "xy" , y = "yx" , z = "x" in w = "yy"
149	b x = "xx" , y = "y" , z = "x" in w = "yy"
150	a odvečne vrstice so: 1; izpiše: 10
150	b odvečne vrstice so: 1, 2, 3; izpiše: 1
150	c odvečne vrstice so: brez; izpiše: 3
150	č odvečne vrstice so: 2; izpiše: 1, 3
151	a z; x; y
151	b x; x; x
152	a y
152	b x; x
152	c x; x; y ali x; y; x
153	a w; w
153	b x; x
154	a pak, park, par, pir, pik in pika
155	a pi, pie, pier, piece, probe, pro, prat in pray
156	a 13, 15, 16, 17, 19 in 20
156	b -4, -2, -1, 0, 1, 2, 3 in 4
157	a -2, -1, 0, 1, 2, 3, 5, 7 in 8
157	b -1, 0 in 1

Tabela 11.1 – Se nadaljuje

Stran	Rešitev
158	a
158	b
158	c

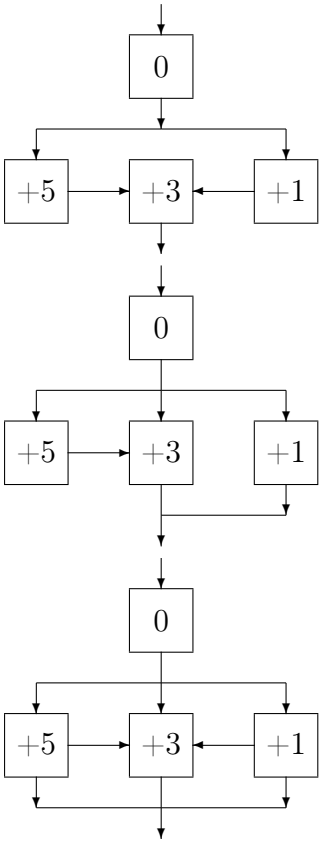
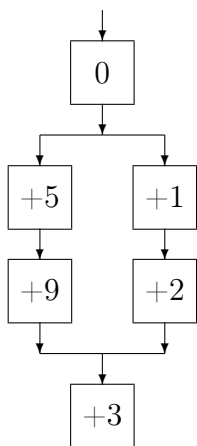




Tabela 11.1 – Se nadaljuje

Stran Rešitev

159 a



159 b

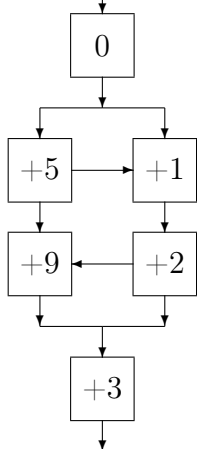


Tabela 11.1 – Se nadaljuje

Stran	Rešitev
160	<div><div>a</div><pre>graph TD; Entry(( )) --&gt; 0_1[0]; 0_1 --&gt; 1_1[+1]; 1_1 --&gt; 5_1[+5]; 5_1 --&gt; 9_1[+9]; 9_1 --&gt; 3_1[+3]; 3_1 --&gt; 0_2[0]; 0_2 --&gt; 1_2[+1]; 1_2 --&gt; 2_2[+2]; 2_2 --&gt; 9_2[+9]; 9_2 --&gt; 3_2[+3]; 3_2 --&gt; Exit(( ))</pre></div>
160	<div><div>b</div><pre>graph TD; Entry(( )) --&gt; 0_1[0]; 0_1 --&gt; 1_1[+1]; 1_1 --&gt; 2_1[+2]; 2_1 --&gt; 9_1[+9]; 9_1 --&gt; 3_1[+3]; 3_1 --&gt; Exit(( ))</pre></div>

Tabela 11.1 – Se nadaljuje

Stran	Rešitev
161	a
	<pre>graph TD; Start(( )) --&gt; N0[0]; N0 --&gt; N5[+5]; N0 --&gt; N1[+1]; N5 --&gt; N9[+9]; N1 --&gt; N2[+2]; N9 --&gt; N2; N2 --&gt; N3[+3]; N1 --&gt; N3; N3 --&gt; End1(( ))</pre>
161	b
162	a 200
162	b 15
162	c 30
162	č 0
162	d 15

Tabela 11.1 – Se nadaljuje

Stran	Rešitev
163	a pravočasno
163	b pravočasno
163	c karambol 4,9
163	č za las
164	a $a < 10$
164	b $a < 10$ ali $a > -5$
164	c $a == 0$ ali $a \geq 10$
164	č $a \geq 10$ ali $a > -5$
164	d $a < 10$
164	e $a == 0$ ali $a < 10$
165	a $a < c$ ali $a < b$
165	b $b == c$ ali $a > b$
165	c $b == c$ ali $a > b$
165	č $a < b$
165	d $a > b$
165	e $a < b$ , $a < c$ ali $b == c$
166	a $(a == 8, a > b)$
166	b $(a < 4, a > b)$
166	c $(a < 4, a == 8)$ ali $(a < 4, a < 4)$
166	č $(a < 4, a == 8)$ , $(a < 4, a < 4)$ , $(a == 8, a < 4)$ ali $(a == 8, a == 8)$
166	d $(a == 8, a > b)$ ali $(a < 4, a > b)$
167	a $(a < 3, a > 3)$
167	b $(a == a, a > 3)$
167	c $(a < 3, \text{ne vpliva})$ ali $(a > 3, \text{ne vpliva})$
167	č $(a == a, a < 3)$ ali $(a > 3, a < 3)$
167	d $(a == a, a < 3)$

Tabela 11.1 – Se nadaljuje

Stran	Rešitev						
	<table> <tr> <th>Oznaka vejitve</th><th>Pogoj</th></tr> <tr> <td>168 a</td><td> <math>p_1 \rightarrow a &lt; 3</math>  <math>p_2 \rightarrow a &gt; 6</math>  <math>p_3 \rightarrow a &gt; 8</math> </td></tr> <tr> <th>Oznaka vejitve</th><th>Pogoj</th></tr> </table>	Oznaka vejitve	Pogoj	168 a	$p_1 \rightarrow a < 3$ $p_2 \rightarrow a > 6$ $p_3 \rightarrow a > 8$	Oznaka vejitve	Pogoj
Oznaka vejitve	Pogoj						
168 a	$p_1 \rightarrow a < 3$ $p_2 \rightarrow a > 6$ $p_3 \rightarrow a > 8$						
Oznaka vejitve	Pogoj						
168 b	<table> <tr> <th>Oznaka vejitve</th><th>Pogoj</th></tr> <tr> <td> <math>p_1 \rightarrow a &lt; 3</math>  <math>p_2 \rightarrow a &gt; 6</math>  <math>p_3 \rightarrow a &gt; 8</math> </td><td></td></tr> <tr> <th>Oznaka vejitve</th><th>Pogoj</th></tr> </table>	Oznaka vejitve	Pogoj	$p_1 \rightarrow a < 3$ $p_2 \rightarrow a > 6$ $p_3 \rightarrow a > 8$		Oznaka vejitve	Pogoj
Oznaka vejitve	Pogoj						
$p_1 \rightarrow a < 3$ $p_2 \rightarrow a > 6$ $p_3 \rightarrow a > 8$							
Oznaka vejitve	Pogoj						
168 c	<table> <tr> <th>Oznaka vejitve</th><th>Pogoj</th></tr> <tr> <td> <math>p_1 \rightarrow a &lt; 3</math>  <math>p_2 \rightarrow a &gt; 6</math>  <math>p_3 \rightarrow a &gt; 8</math> </td><td></td></tr> <tr> <th>Oznaka vejitve</th><th>Pogoj</th></tr> </table>	Oznaka vejitve	Pogoj	$p_1 \rightarrow a < 3$ $p_2 \rightarrow a > 6$ $p_3 \rightarrow a > 8$		Oznaka vejitve	Pogoj
Oznaka vejitve	Pogoj						
$p_1 \rightarrow a < 3$ $p_2 \rightarrow a > 6$ $p_3 \rightarrow a > 8$							
Oznaka vejitve	Pogoj						
169 a	<table> <tr> <th>Oznaka vejitve</th><th>Pogoj</th></tr> <tr> <td> <math>p_1 \rightarrow a &gt; 3</math>  <math>p_2 \rightarrow a == 5</math>  <math>p_3 \rightarrow a &lt; 14</math>  <math>p_4 \rightarrow a &lt; 5</math> </td><td></td></tr> <tr> <th>Oznaka vejitve</th><th>Pogoj</th></tr> </table>	Oznaka vejitve	Pogoj	$p_1 \rightarrow a > 3$ $p_2 \rightarrow a == 5$ $p_3 \rightarrow a < 14$ $p_4 \rightarrow a < 5$		Oznaka vejitve	Pogoj
Oznaka vejitve	Pogoj						
$p_1 \rightarrow a > 3$ $p_2 \rightarrow a == 5$ $p_3 \rightarrow a < 14$ $p_4 \rightarrow a < 5$							
Oznaka vejitve	Pogoj						
169 b	<table> <tr> <th>Oznaka vejitve</th><th>Pogoj</th></tr> <tr> <td> <math>p_1 \rightarrow a &gt; 3</math>  <math>p_2 \rightarrow a == 5</math>  <math>p_3 \rightarrow a &lt; 14</math>  <math>p_4 \rightarrow a &lt; 5</math> </td><td></td></tr> <tr> <th>Oznaka vejitve</th><th>Pogoj</th></tr> </table>	Oznaka vejitve	Pogoj	$p_1 \rightarrow a > 3$ $p_2 \rightarrow a == 5$ $p_3 \rightarrow a < 14$ $p_4 \rightarrow a < 5$		Oznaka vejitve	Pogoj
Oznaka vejitve	Pogoj						
$p_1 \rightarrow a > 3$ $p_2 \rightarrow a == 5$ $p_3 \rightarrow a < 14$ $p_4 \rightarrow a < 5$							
Oznaka vejitve	Pogoj						

Tabela 11.1 – Se nadaljuje

Stran Rešitev

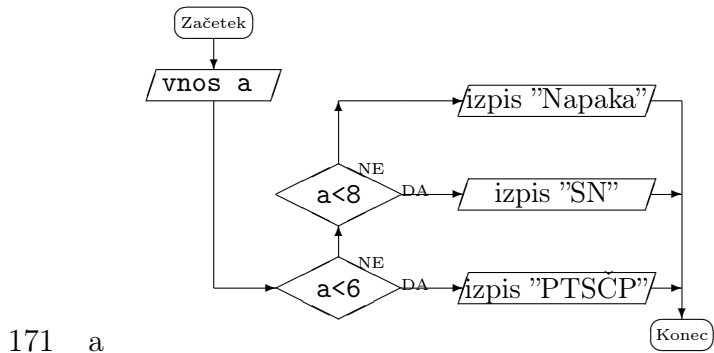
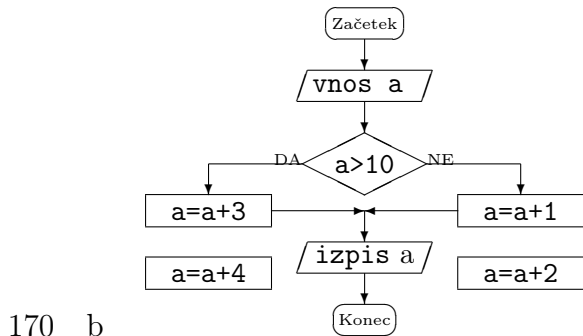
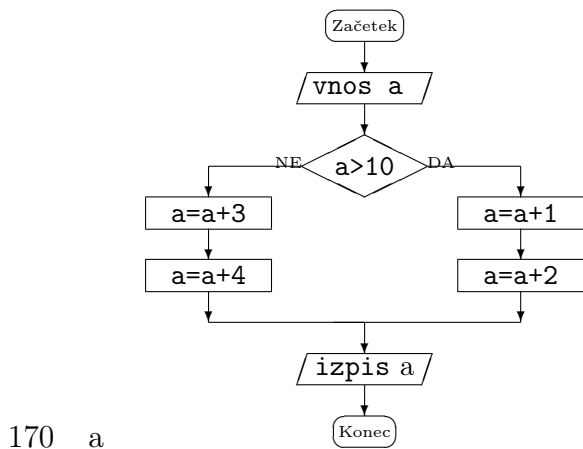
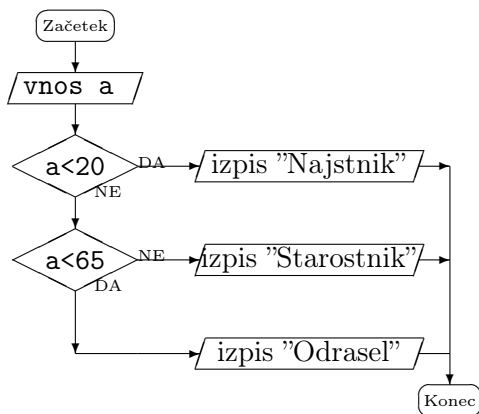


Tabela 11.1 – Se nadaljuje

Stran Rešitev

171 b



172 a

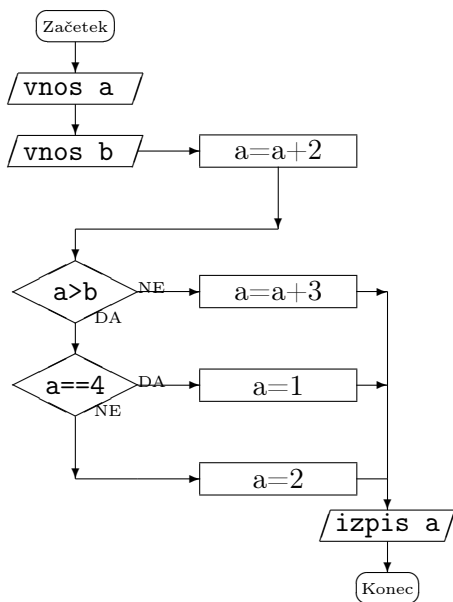
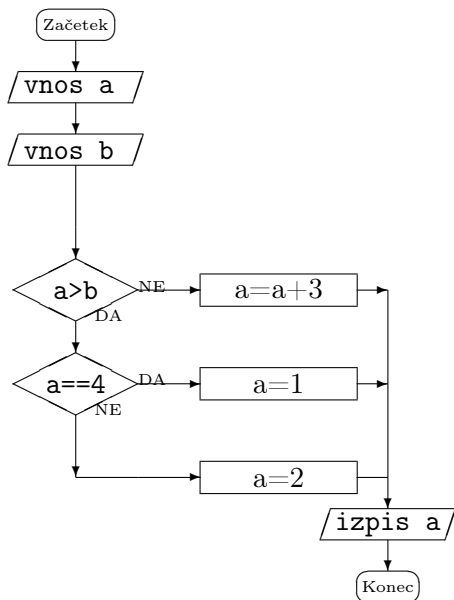


Tabela 11.1 – Se nadaljuje

---

Stran    Rešitev

---

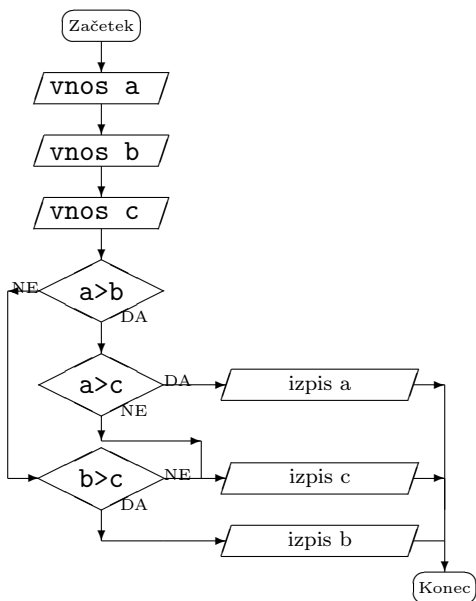


173    a



Tabela 11.1 – Se nadaljuje

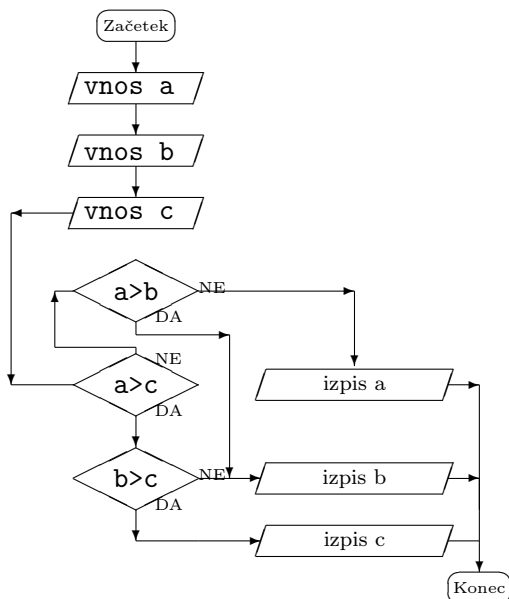
Stran Rešitev



174 a

Tabela 11.1 – Se nadaljuje

Stran Rešitev



175 a  
 176 a 2\*  
 176 b 2\*4\*  
 176 c 2\*4\*6\*  
 176 č 2\*4\*6\*8\*10\*  
 176 d 2\*4\*6\*8\*10\*12\*14\*  
 177 a 2\*1\*  
 177 b 6\*5\*4\*3\*2\*1\*  
 177 c 3\*2\*1\*  
 177 č -1\*  
 177 d -4\*  
 178 a 2\*1\*K1\*K  
 178 b 3\*2\*1\*K2\*1\*K1\*K  
 178 c -1\*K  
 178 č 4\*3\*2\*1\*K3\*2\*1\*K2\*1\*K1\*K

Tabela 11.1 – Se nadaljuje

---

Stran	Rešitev
178	d 1*K
179	a a==8
179	b a>8
179	c a>8 ali a<>8
179	č a>3
179	d a>3
179	e a==3
180	a a>3
180	b a>3
180	c a==8
180	č a>8
180	d a>8
180	e a==8
181	a a<4
181	b a<>b
181	c a>b
181	č b<4 ali a<>b
181	d a<4
181	e a==1
182	a (a>3, a>7)
182	b (a>7, a>3)
182	c (a>3, a>3) ali (a>3, a>7)
182	č (b==3, a>7)
182	d (a>7, a>7) ali (a>7, b==3)

---

Tabela 11.1 – Se nadaljuje

Stran Rešitev

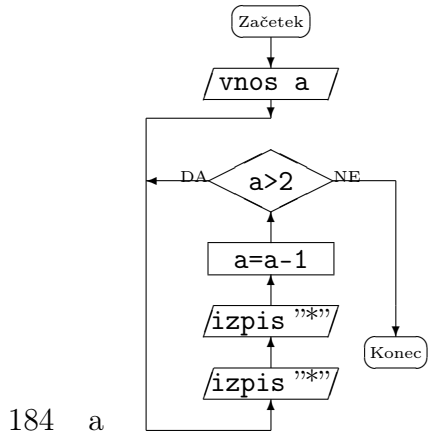
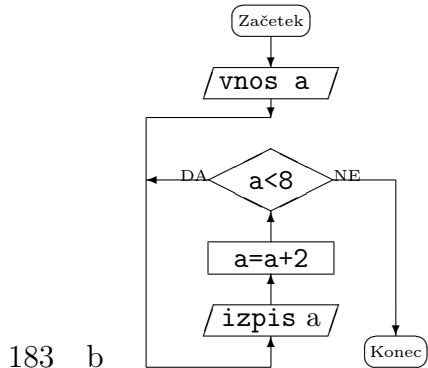
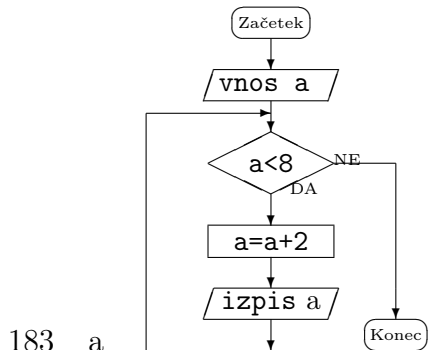
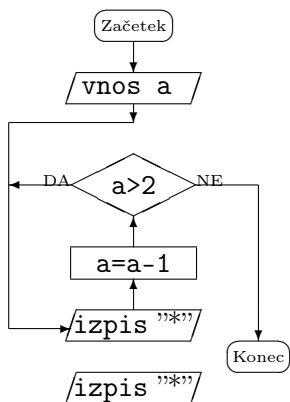
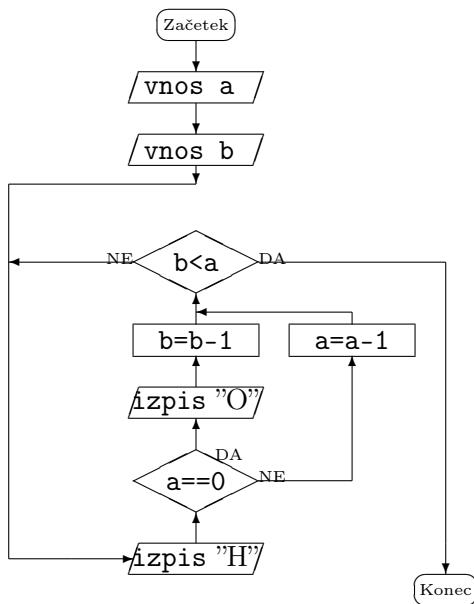


Tabela 11.1 – Se nadaljuje

Stran Rešitev



184 b

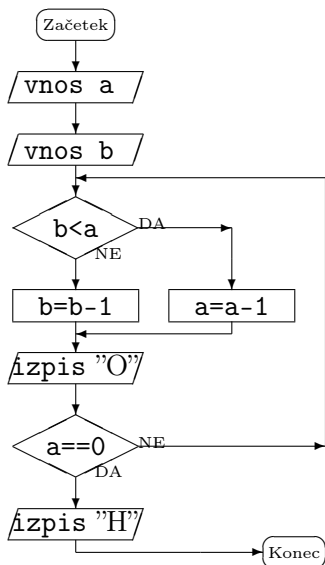


185 a

Tabela 11.1 – Se nadaljuje

Stran Rešitev

185 b



186 a

