

# PIŠEK 2020/21

3. poskusno (spletno) tekmovanje

Naloge in rešitve

Februar 2021

Izbor, priredba in preoblikovanje tekmovalnih nalog: Programski svet tekmovanja

Razvoj tekmovalnega sistema: ACM in FMF v sodelovanju s France-IOI

# KAZALO NALOG

## Tekmovalna kategorija: OŠ 4. – 6. r. ZAČETNIKI

PIŠEK ZOBA ZRNA.....	1
LADJA REŠI BRODOLOMCA.....	2
NEZNANI ELEMENT .....	3
TEKAČ.....	4
SKRITI TARTUFI.....	6

## Tekmovalna kategorija: OŠ 4. – 6. r. NAPREDNI

ZMAJČEK IN CEKINI.....	7
PIŠEK GRE ČEZ CESTO .....	8
ROBOT POTUJE.....	10
VEVERICA NABIRA ŽELOD .....	12
MARTIN V LABIRINTU .....	14

## Tekmovalna kategorija: OŠ 7. – 9. r. ZAČETNIKI

DOSTAVA PAKETOV.....	16
SKRITI TARTUFI.....	18
KRT USTVARJA.....	19
VEVERICA IN IZGUBLJENI LEŠNIKI .....	20
UBBI DUBBI .....	22

## Tekmovalna kategorija: OŠ 7. – 9. r. NAPREDNI

PIŠEK POBIRA PETICE.....	24
PIŠEK IN SEF .....	25
UČITELJ GROZNI .....	27
SNEŽAK RIŠE SNEŽINKE.....	29
JEŽEK POSPRAVLJA.....	31

## Tekmovalna kategorija: SŠ ZAČETNIKI

KEMIK ŽIGA .....	34
BOŽIČKOVA OKRASITEV .....	36
ZAKLAD CEKINOV .....	39

SNEŽKO SNEŽAK IGRA TRI V VRSTO .....	42
ŽELVICA KAJA .....	45

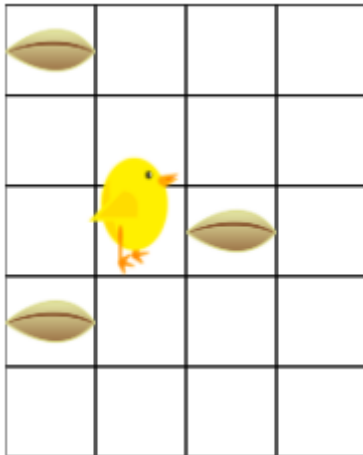
**Tekmovalna kategorija: SŠ NAPREDNI**

PROFESOR GROZNI.....	48
KRTOVA GLEDALIŠKA PREDSTAVA.....	51
MARTIN V LABIRINTU .....	54
DEMOKRATIČNO ČIŠČENJE .....	56
KOŠARKAŠKI TRENING .....	59

## PIŠEK ZOBA ZRNA

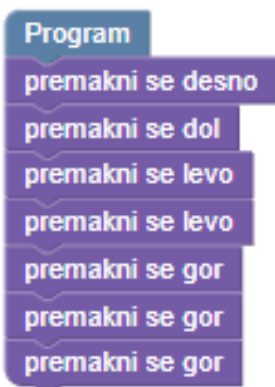
OŠ 4. – 6. r. ZAČETNIKI

Cel dopoldan je Pišek raziskoval svojo okolico in se družil s prijatelji, zato je spet lačen. V njegovi okolici je nekaj zrn. Popelji Piška tako, da bo pozobal vsa zrna. Pišek je zelo lačen, zato pozoba zrna takoj, ko jih najde. Možnih je več rešitev.



[Povezava do naloge](#)

### Rešitev



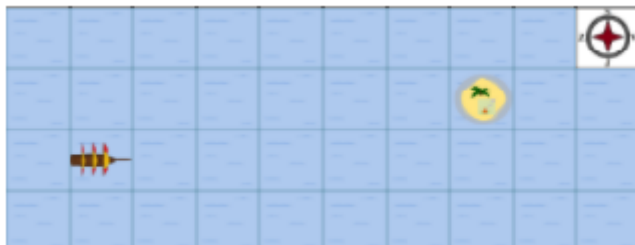
### Ideja reševanja

Delčke uredimo v pravilno zaporedje ukazov premikanja. Zaradi omejitve delčkov, poskusimo poiskati čim krajšo pot. Možnih je več rešitev.

## LADJA REŠI BRODOLOMCA

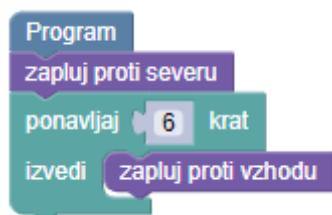
OŠ 4. – 6. r. ZAČETNIKI

Ladja pluje skozi ocean. Posadka opazi dimni signal na bližnjem otoku. Usmeri kapitana ladje tako, da bo rešil brodolomca na otoku.



### Povezava do naloge

## Rešitev



### Ideja reševanja

Ugotovimo, da je ladja oddaljena od otoka za eno polje proti severu in šest polj proti vzhodu. Cilj bi lahko dosegla tako, da bi zgolj nizali delčke enega za drugim. Ker pa se šestkrat ponovi isti delček `zapluj proti vzhodu`, lahko uporabimo preprosto enojno zanko. Delček `zapluj proti severu` pa lahko uporabimo pred zanko ali pa za njo. V obeh primerih bo ladja prišla do otoka, a po drugi poti. Zaradi omejitve delčkov v nalogi, je možna le rešitev z zanko.

## NEZNANI ELEMENT

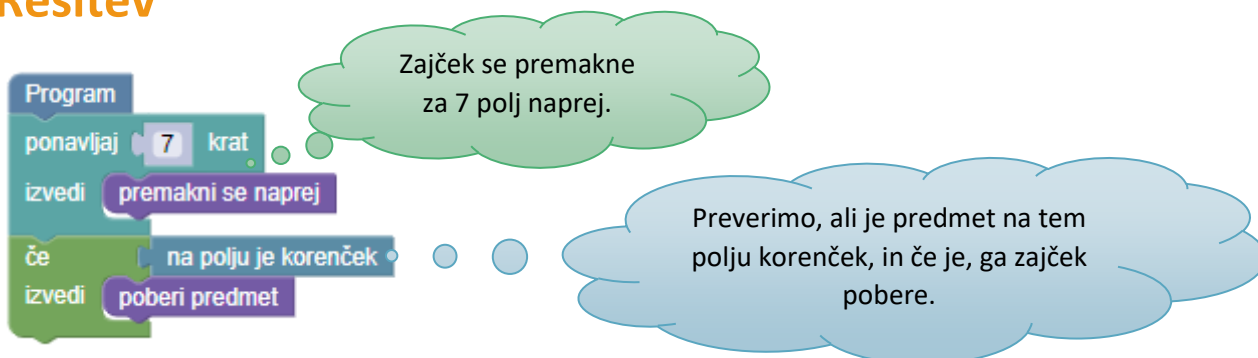
OŠ 4. – 6. r. ZAČETNIKI

Zajček na koncu svoje poti naleti na nek predmet. Če je to korenček, naj ga zajček pobere, drugače pa ne! Pazi, naloga ima več testov.



### Povezava do naloge

### Rešitev



### Ideja reševanja

Najprej si dobro ogledamo vse tri testne primere. Ugotovimo, da se mora zajček vedno premakniti naprej za sedem polj, da pride do predmeta. Lahko bi zgolj nizali delčke, vendar je hitreje, če uporabimo preprosto zanko. Ker je število delčkov omejeno, moramo zanko nujno uporabiti, saj naloge drugače ne moremo rešiti. Na zadnjem polju moramo uporabiti pogojni stavek; in sicer preverjamo, ali je na tem polju korenček, in če je, ga zajček pobere.

## TEKAČ

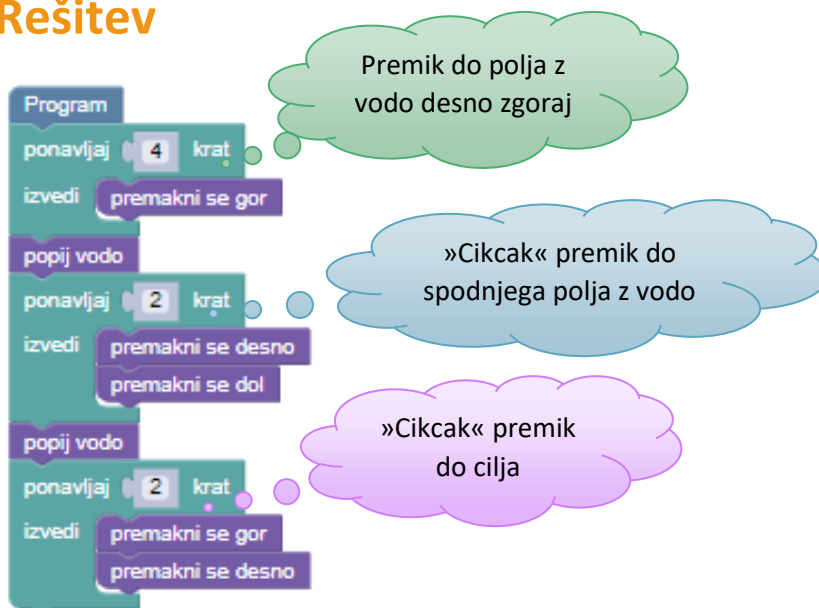
OŠ 4. – 6. r. ZAČETNIKI

Tekač mora priteči do zelenega polja, ki je njegov cilj. Vmes se mora ustaviti na obeh modrih poljih in popiti vodo. Pomagaj priteči tekaču do cilja. Pazi, število dečkov je omejeno.



### Povezava do naloge

## Rešitev



## Ideja reševanja

Tekača premikamo z zaporedjem ukazov, ko pridemo na polje z vodo, jo tekač popije. Zaradi omejitve delčkov, moramo poiskati čim krajšo pot. Hitro lahko ugotovimo, da lahko do polja z vodo v zgornjem desnem kotu pridemo z uporabo prve zanke v našem programu.

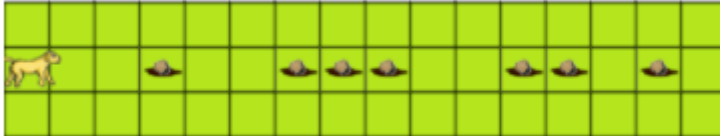


Nato pa sledi težji premislek: do spodnje plastenke z vodo bi najhitreje prišli kar po diagonali, a tega delčka nimamo na voljo. Lahko bi se premaknili dve polji desno in dve polji dol, a bi nam v tem primeru zmanjkalo delčkov za končanje programa. Edina možnost, ki nam ostane je, da se premikamo »cikcak«, torej eno polje desno in eno polje navzdol. Ker moramo to gibanje ponoviti dvakrat, lahko uporabimo zanko. Premik do cilja je zelo podoben prejšnjemu, zato znova uporabimo zanko, tokrat pa se premikamo »cikcak« desno in gor.

## SKRITI TARTUFI

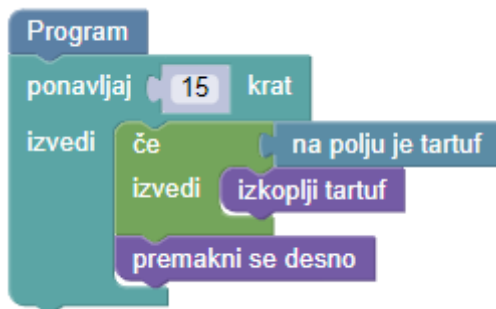
OŠ 4. – 6. r. ZAČETNIKI

Radovedni kužek se je odpravil iskat tartufe, pomagaj mu jih odkopati. Z ukazom "če" lahko preveriš ali so na določenem polju zares zakopani tartufi.



### Povezava do naloge

## Rešitev



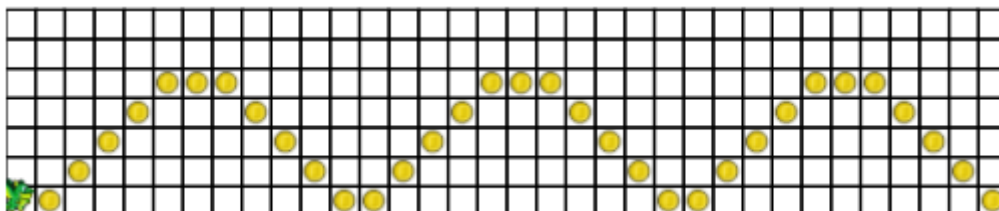
### Ideja reševanja

Nalogo bi lahko rešili s preprostim zaporedjem delčkov `premakni se desno` in `izkoplji tartuf`, ko bi kuža bil na polju s tartufom. Omejitev delčkov pa od nas zahteva poznavanje in uporabo senzorjev. Z uporabo pogojnega stavka `če` in senzorja `na polju je tartuf` preverimo, ali se na polju nahaja tartuf, in če se, ga kuža izkoplje, nato pa se premakne naprej. Ker mora kuža ta postopek ponoviti na vsakem od naslednjih 15 polj, uporabimo preprosto enojno zanko.

## ZMAJČEK IN CEKINI

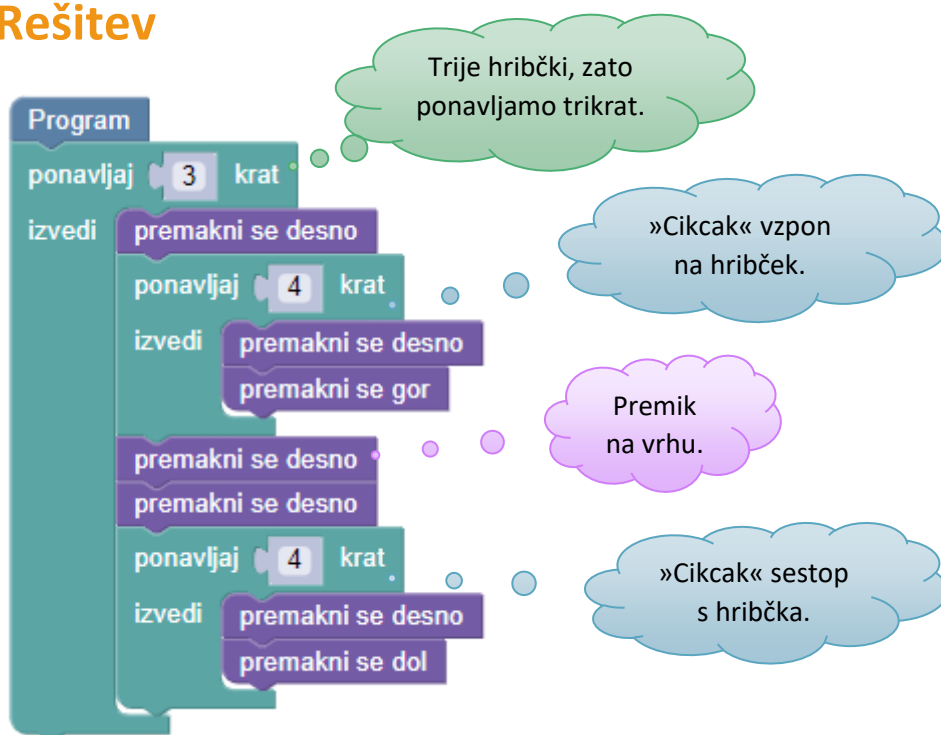
OŠ 4. – 6. r. NAPREDNI

Zmajček je med letenjem sem in tja izgubil svoje cekine. Napiši program, s pomočjo katerega jih bo spretno pobral, preden jih odkrije kdo drug ...



### Povezava do naloge

### Rešitev



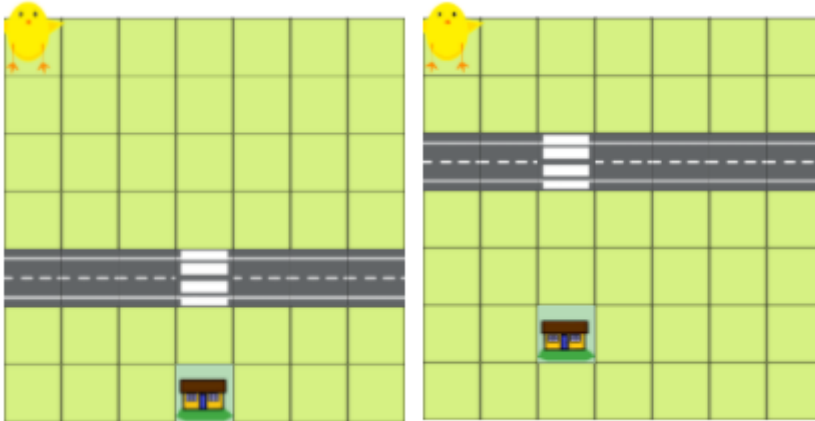
### Ideja reševanja

Najprej poiščemo vzorec, ki se ponavlja. Vidimo tri hribčke, zato vemo, da bomo lahko uporabili zanko, ki bo trikrat ponovila pot enega hribčka. Vsak hribček pa je sestavljen iz vzpona, vrha in sestopa. Za vzpon in sestop bomo uporabili zanko, ki bo zmajčka premikala »cikcak« po diagonali navzgor oz. navzdol.

## PIŠEK GRE ČEZ CESTO

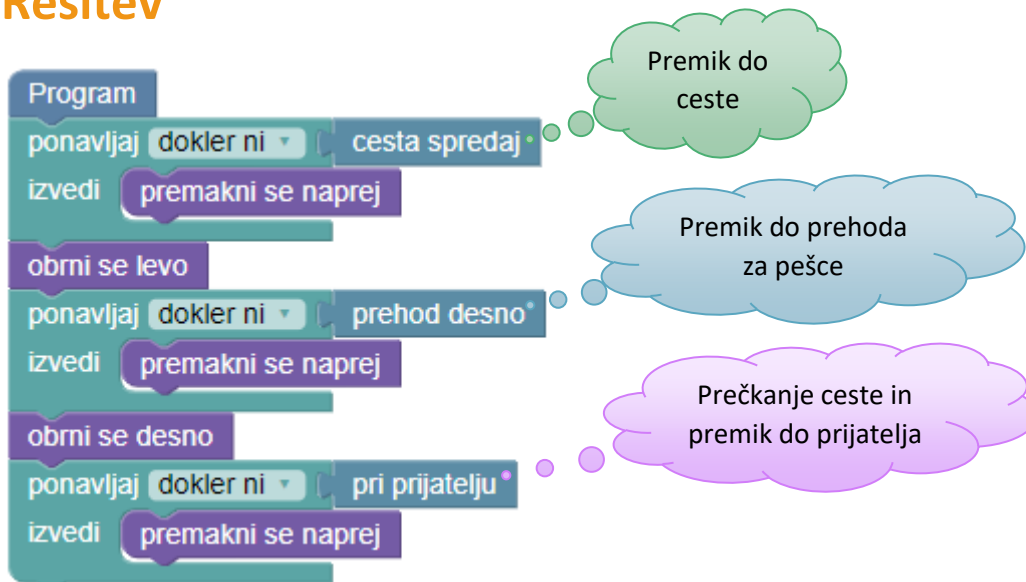
OŠ 4. – 6. r. NAPREDNI

Pišek se odpravi k prijatelju, ki stanuje na drugi strani ceste. Najprej mora poiskati prehod za pešce in varno prečkati cesto, nato pa je v nekaj korakih pri prijatelju. Napiši program, ki bo Piška pripeljal do prijatelja, a pazi, prehod za pešce je lahko na različnih mestih.



### Povezava do naloge

### Rešitev



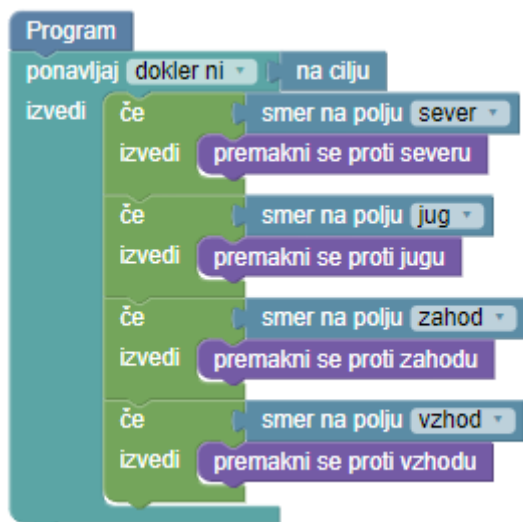
## Ideja reševanja

Najprej si dobro ogledamo oba testna primera. Problem lahko razbijemo na več podproblemov. 1) Pišek mora najprej priti do ceste, ki pa je od njega lahko različno oddaljena. Ta problem rešimo tako, da uporabimo zanko `dokler ni` in senzor `cesta spredaj`. 2) Ko Pišek pride do ceste, mora najti prehod za pešce. Vidimo, da se mora obrniti v levo, nato pa se bo premikal naprej, dokler ne bo na njegovi levi prehod, kar dosežemo z uporabo druge zanke `dokler ni` in »`prehod desno`. 3) Vidimo, da je prijateljev dom v istem stolpcu kot prehod za pešce, kar pomeni, da ko je Pišek ob prehodu, se obrne desno in se premika naprej dokler ni pri prijatelju, kar naredimo s tretjo zanko v kodi in senzorjem `pri prijatelju`.

## OŠ 4. – 6. r. NAPREDNI

[illegible]

## Rešitev



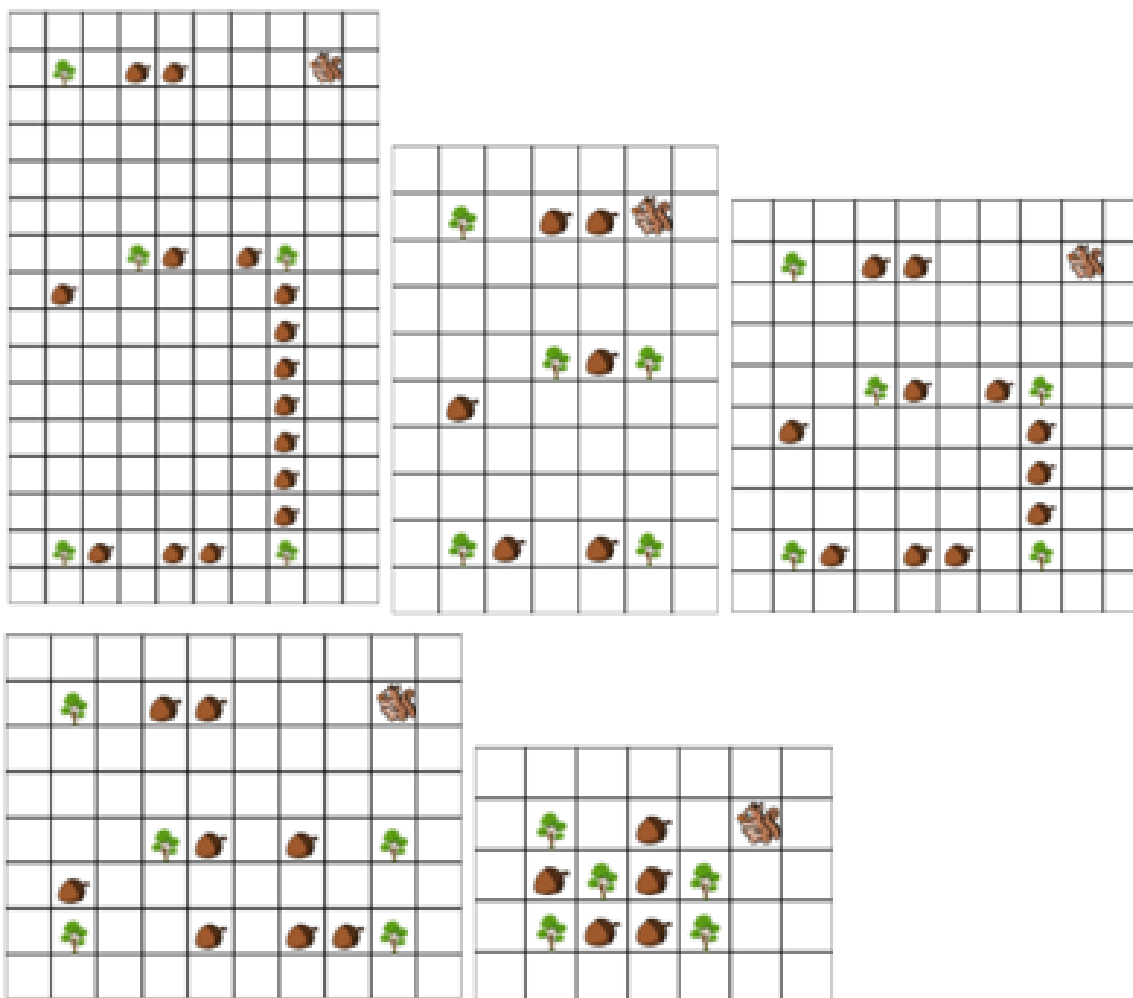
## Ideja reševanja

Najprej si dobro ogledamo oba testna primera. Vidimo, da sta poti, ki jih mora robot prehoditi precej različni, zato moramo poiskati univerzalno rešitev. Premikanje v vseh štirih smereh neba določimo z uporabo pogojnih stavkov. Uporabiti moramo senzorje, s pomočjo katerih robot preveri smer, v katero kaže puščica na polju. Z uporabo pogojnih stavkov pa se robot premakne v pravo smer. Ker sta dolžini poti v obeh testnih primerih različni, ne moremo uporabiti preproste zanke, lahko pa uporabimo zanko `dokler`, s katero program ponavlja premikanje v smeri puščic vse dokler robot ni na cilju.

## VEVERICA NABIRA ŽELOD

OŠ 4. – 6. r. NAPREDNI

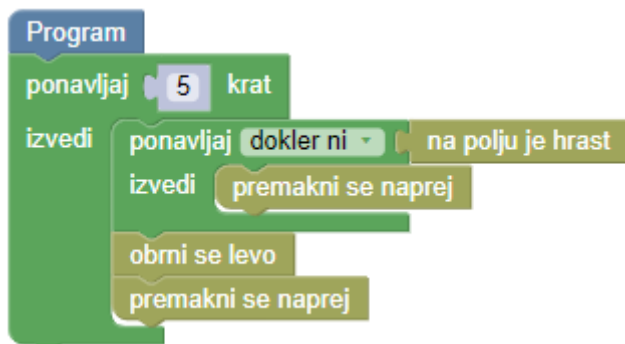
Veverica nabira želod in ugotovi, da ga lažje najde, če sledi hrastom. Vsakič, ko pride do hrasta, zamenja smer gibanja. Poskusi poiskati vzorec, ki je skupen vsem testnim primerom. Da ugotoviš, ali je na polju hrast, uporabi orodjarno senzorji.



[Povezava do naloge](#)



## Rešitev



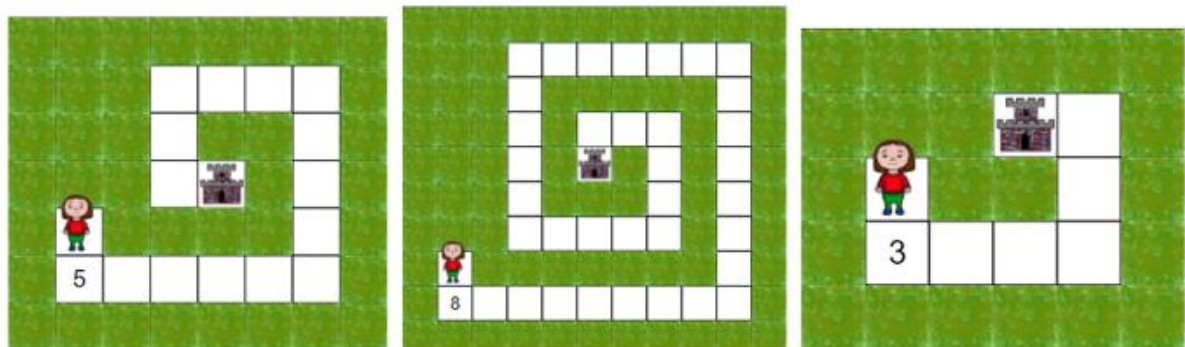
## Ideja reševanja

Najprej si dobro ogledamo vse testne primere, ki so si na prvi pogled precej različni. Opazimo dve stvari: 1) ko veverica pride do hrasta, se mora vedno obrniti levo, če želi priti do naslednjega želoda in 2) vsak testni primer ima natanko 5 hrastov. Uporabimo torej dvojno zanko, pri čimer je notranja zanka tipa `ponavlja dokler`, delček `na polju je hrast` pa najdemo med senzorji.

## MARTIN V LABIRINTU

OŠ 4. – 6. r. NAPREDNI

Pomagaj Martinu najti pot do gradu, kjer ga čaka njegova princesa. Grad je skrit globoko v labirintu, ki pa je lahko različno velik. Na začetku labirinta, je vedno polje, ki pove, dolžino najdaljše stranice, nato pa se labirint zavija kot kača, in ima vsako stranico za eno polje krajšo. Napiši program, ki bo Martina pripeljal do gradu ne glede na velikost labirinta.



### Povezava do naloge

## Rešitev

### 1. Rešitev

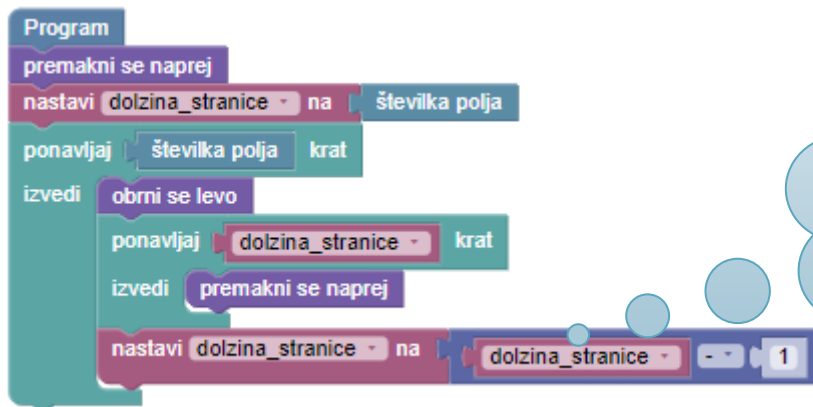


Zanka  
'for' (za)

Spremenljivko *i* v vsaki ponovitvi zanke zmanjšamo za 1, ker je vsaka naslednja stranica labirinta krajša za 1 polje.

Prvič se bo Martin premaknil naprej za največ polj, torej toliko kot je številka polja, zato bo vrednost spremenljivke *i*, v prvi izvedbi zanke največja.

## 2. Rešitev



Spremenljivko *dolzina\_stranice* v vsaki ponovitvi zanke zmanjšamo za 1, ker je vsaka naslednja stranica labirinta krajša za 1 polje.

### Ideja reševanja

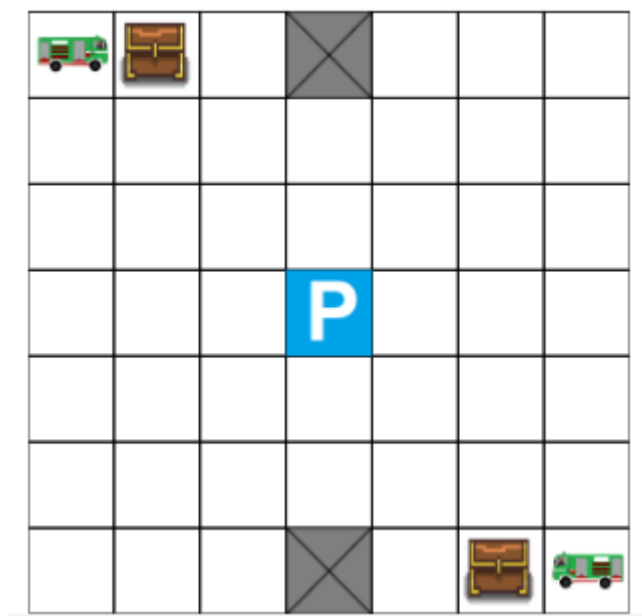
Da bomo lahko rešili nalogo, si moramo najprej dobro ogledati vse tri testne primere in poiskati vzorec. Najprej se mora Martin torej premakniti naprej za eno polje. Potem pa bo ponavljal isti vzorec tolikokrat, kolikor je številka na tem polju. In sicer se bo najprej obrnil levo, nato pa se bo premikal naprej. A po vsaki ponovitvi se bo Martin premaknil naprej za eno polje manj. Uporabiti moramo torej zanko `for` (za), ki bo za vsako vrednost spremenljivke *i* izvedla premik v levo, nato pa še za število *i* premikov naprej. S to zanko računalniku naročimo, da naj vrednost spremenljivke *i* najprej nastavi na vrednost `številka polja`, po vsaki ponovitvi pa naj jo zmanjša za 1 (`s korakom 1`), zanko pa naj ponavlja, dokler je *i* večji od 1.

Nalogo lahko rešimo tudi brez uporabe zanke `for`, in sicer tako, da ustvarimo novo spremenljivko in jo spreminjamo tako, kot bi to delala zanka `for` namesto nas. Najprej nastavimo vrednost spremenljivke *dolzina\_stranice* na vrednost `številke polja`. Ker vemo, da labirint naredi toliko zavojev, kot je številka polja, bomo uporabili preprosto zanko, ki se bo izvedla prav tolikokrat. Martin se bo torej vsakič obrnil levo, nato pa premaknil naprej, prvič za vrednost, ki smo jo že shranili v spremenljivki *dolzina\_stranice*, vsakič naslednjič pa za eno polje manj. Zato moramo vrednost spremenljivke v vsaki ponovitvi zanke zmanjšati za ena. Možne so tudi drugačne rešitve.

## DOSTAVA PAKETOV

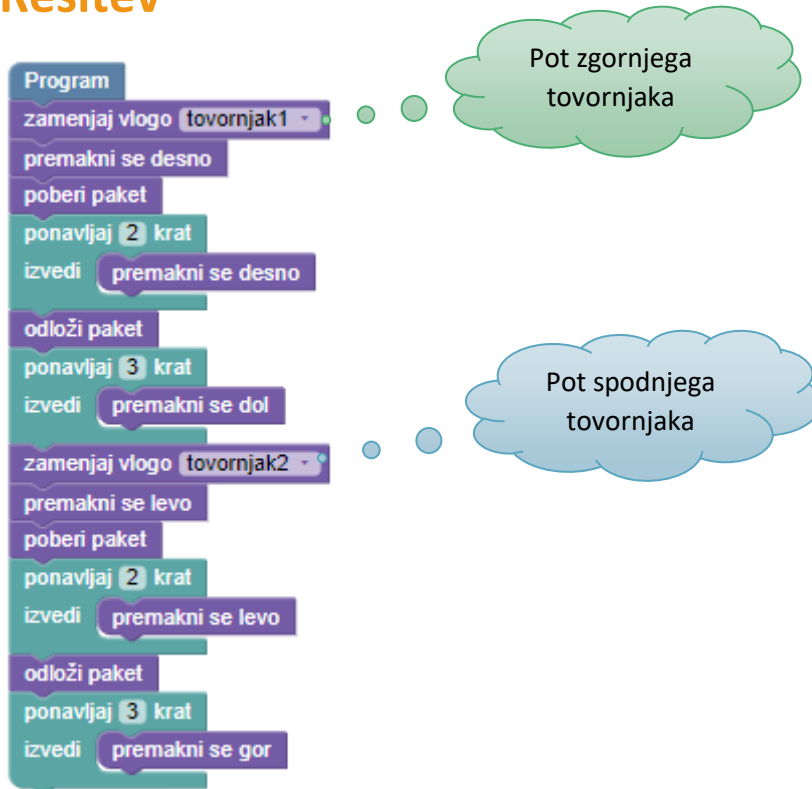
OŠ 7. – 9. r. ZAČETNIKI

Nekje na gradbišču se nahajata dva sumljiva paketa. Pomagaj voznikoma pri iskanju paketov. Ko voznik najde paket, naj ga naloži na tovornjak in odpelje do najbližje varne cone (sivo polje). Tam naj paket odloži in se odpelje v garažo (parkirišče). Na koncu se mora posamezen paket nahajati v najbližji varni coni, tovornjaka pa morata biti parkirana v garaži.



[Povezava do naloge](#)

## Rešitev



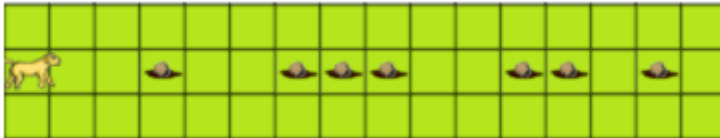
## Ideja reševanja

Vidimo, da moramo premikati oba tovarnjaka. Najprej nastavimo vlogo zgornjemu tovarnjaku, ki je `tovornjak1`. To lahko preprosto preizkusimo z zagonom programa z enim premikom oz. lahko ugotovimo, da prvega delčka niti ne potrebujemo, ker računalnik privzame, da je prvi tovarnjak zgornji. Izvedemo premike za zgornji tovarnjak. Uporabiti moramo eno zanko za tri premike dol proti parkirišču, saj nam bo v nasprotnem primeru zmanjkalo delčkov za dokončanje programa. Nato zamenjamo vlogo tovarnjaka in ponovimo zrcalni postopek še za spodnji tovarnjak.

## SKRITI TARTUFI

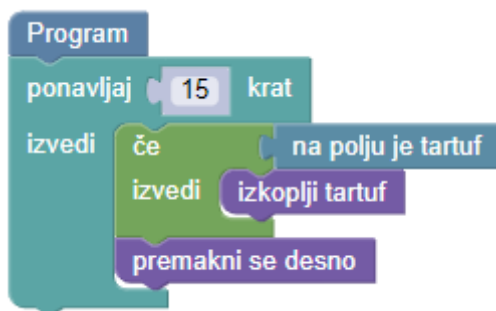
OŠ 7. – 9. r. ZAČETNIKI

Radovedni kužek se je odpravil iskat tartufe, pomagaj mu jih odkopati. Z ukazom “če” lahko preveriš ali so na določenem polju zares zakopani tartufi.



### Povezava do naloge

## Rešitev



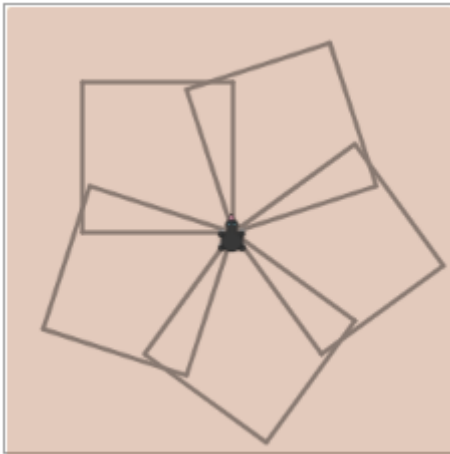
### Ideja reševanja

Nalogo bi lahko rešili s preprostim zaporedjem delčkov `premakni se desno` in `izkoplji tartuf`, ko bi bil kuža na polju s tartufom. Omejitev delčkov pa od nas zahteva poznavanje in uporabo senzorjev. Z uporabo pogojnega stavka `če` in senzorja `na polju tartuf` preverimo, ali se na polju nahaja tartuf, in če se, ga kuža izkoplje, nato pa se premakne naprej. Ker mora kuža ta postopek ponoviti na vsakem od naslednjih 15 polj, uporabimo preprosto enojno zanko.

## KRT USTVARJA

OŠ 7. – 9. r. ZAČETNIKI

Krt je znan kot izjemen umetnik, ki s svojimi krtinami ustvarja znane likovne predstave za vse prebivalce travnika. Danes želi narediti obliko vetrnice. Dolgo je risal načrte in ugotovil, da lahko vetrnico naredi iz petih kvadratov. Po vsakem narisanim kvadratu pa se mora obrniti za 72 stopinj. Ker se je pod zemljo težko orientirati, si je napisal navodila, ki pa niso popolna. Pomagaj mu jih popraviti.



### Povezava do naloge

## Rešitev



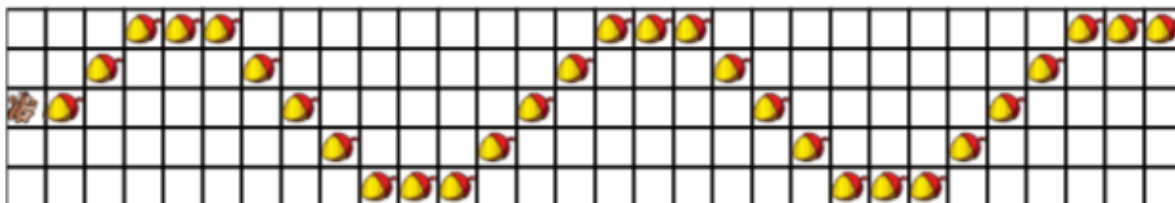
### Ideja reševanja

Program je že napisan vnaprej. Hitro lahko ugotovimo, da notranja zanka riše kvadrat, ker se ponovi štirikrat, toliko kot ima kvadrat stranic. Najprej moramo torej ugotoviti dolžino stranice kvadrata. S poskušanjem pridemo do dolžine 100 premikov. Ker vemo, da ima kvadrat notranje kote prave, torej 90 stopinj, lahko vpišemo tudi to vrednost. Na koncu pa dodamo še obrat za 72 stopinj, kar nam pove že navodilo. Zunanja zanka se ponavlja petkrat, ker krt riše pet kvadratov.

## VEVERICA IN IZGUBLJENI LEŠNIKI

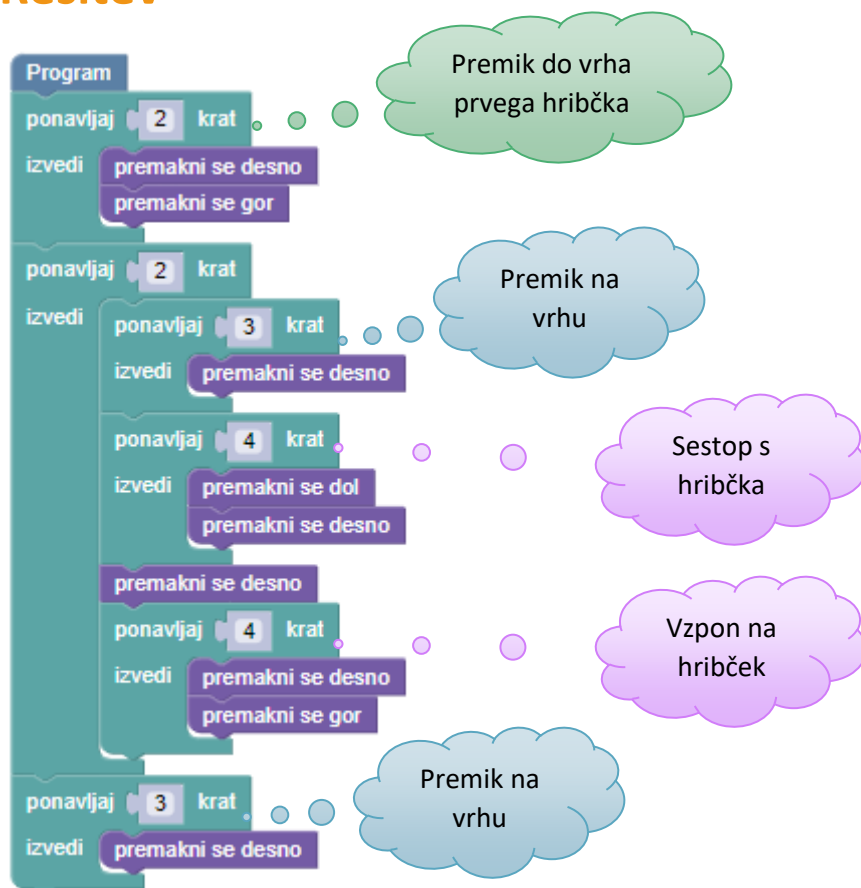
OŠ 7. – 9. r. ZAČETNIKI

Veverica je med nevihto sem in tja izgubila svoje lešnike. Napiši program, ki ji jih bo pomagal pobrati, preden jih odnese kdo drug. Veverica lešnik pobere takoj, ko pride na njegovo polje.



### Povezava do naloge

### Rešitev





## Ideja reševanja

Če si dobro ogledamo testni primer, lahko vidimo, da imamo tri hribčke. Problem razdelimo na več podproblemov in poiščemo del, ki se v kodi ponavlja. Ko pridemo do vrha prvega hribčka s prvo zanko v kodi, se dvakrat ponovi del kode, ki je sestavljen iz delov: »premik na vrhu«, »sestop s hribčka« in »vzpon na hribček«. Na koncu moramo še enkrat ponoviti del »premik na vrhu«. Problem lahko razbijemo tudi na drugačne podprobleme, takrat bo rešitev seveda nekoliko drugačna.

## UBBI DUBBI

OŠ 7. – 9. r. ZAČETNIKI

Piškovi bandi je policija začela prestregati sporočila. Zato so se odločili, da bodo vsa sporočila zapisali v jeziku Ubbi Dubbi. Pomagaj jim napisati program, ki jim bo pomagal prevajati besede v jezik Ubbi Dubbi. Besedo se v jezik Ubbi Dubbi prevede tako, da se pred vsak samoglasnik vrine črki 'ub'.

Input: mačka šola ananas	Output: mubačkuba šuboluba ubanubanubas
-----------------------------------	--

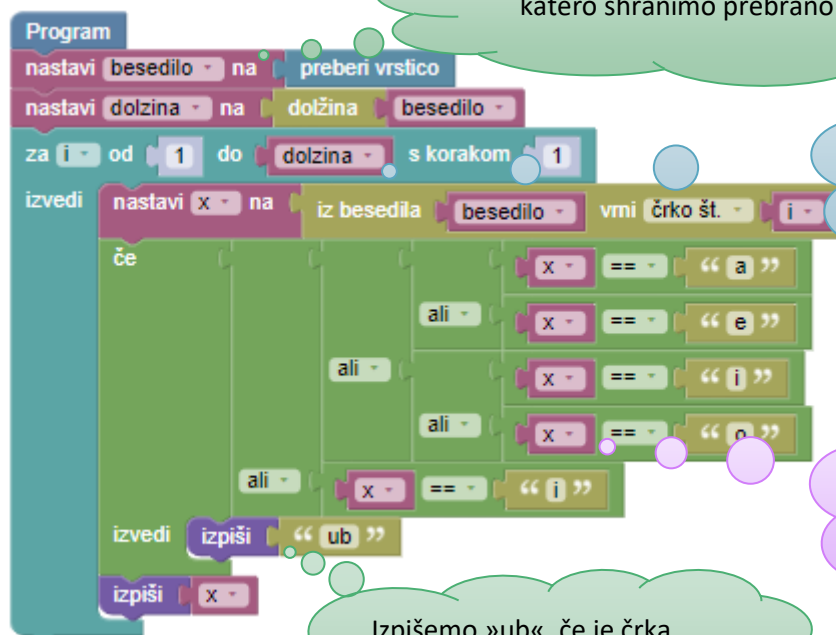
Input: mačka	Output:	Input: šola	Output:

Input: ananas	Output:

[Povezava do naloge](#)

## Rešitev



Ustvarimo novo spremenljivko, v katero shranimo prebrano besedo.

Zanko bomo ponavljali tolikokrat, kolikor črk ima beseda (kolikor je beseda dolga).

Za vsako črko posebej (spremenljivka *x*) preverjamo, ali je samoglasnik.

Izpišemo »ub«, če je črka shranjena v spremenljivki *x*, samoglasnik.

## Ideja reševanja

Ustvariti moramo novo spremenljivko, v katero bomo shranili besedilo na vходу, mi smo jo poimenovali *besedilo*. Nato pa moramo preveriti še dolžino spremenljivke *besedilo*, saj bomo naslednje korake v kodi ponavljali po vrsti za vsako črko besede. To dolžino lahko shranimo v novo spremenljivko, npr. *dolzina*, ali pa jo uporabimo kar direktno v naslednji zanki.

Spremenljivko *x* bomo v vsaki ponovitvi nastavili na naslednjo črko, torej prvič bo to prva črka besede, drugič bo druga itd. Nato pa moramo za vsako črko preveriti, ali je le-ta samoglasnik (a, e, i, o, u). To naredimo z uporabo pogojnega stavka in logičnega znaka *ali*. Kadar je črka samoglasnik, izpišemo »ub«, nato pa še spremenljivko *x*. Zanka se bo ponovila za vsako črko v besedi, in sicer bo po vsaki ponovitvi izpisala pregledano črko, pred samoglasnike pa bo vrnila še »ub«.

## PIŠEK POBIRA PETICE

OŠ 7. – 9. r. NAPREDNI

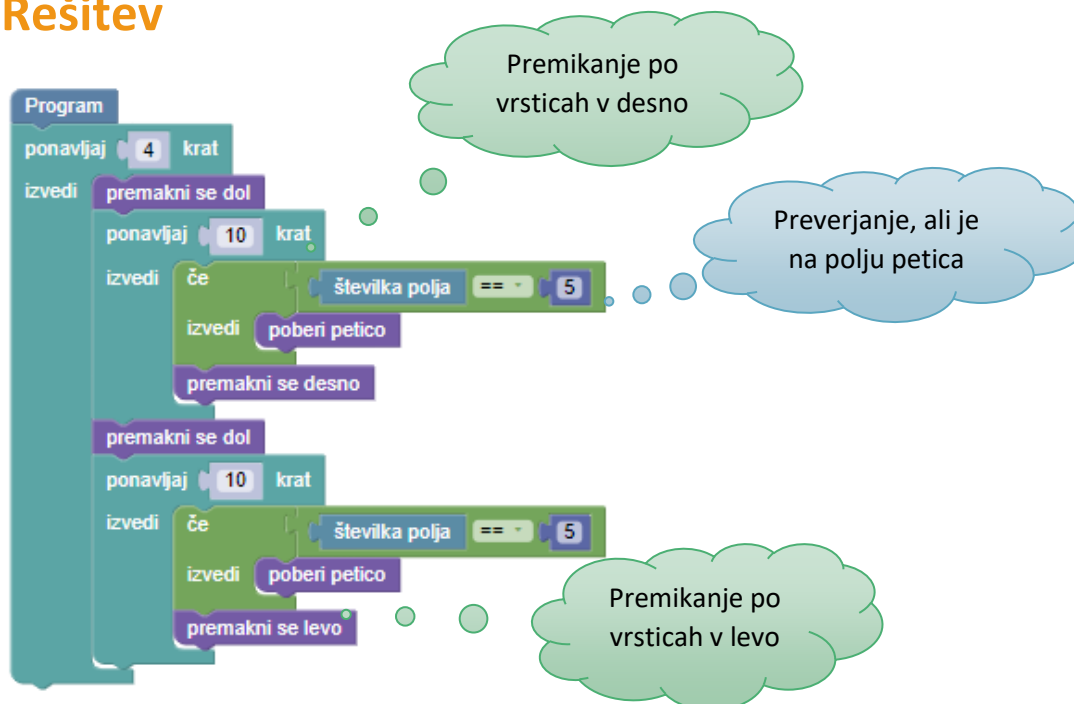
Napiši program, ki Piška vodi po mreži tako, da preskakuje ocene 4 in slabše, petice pa pobere. Pazi, število delčkov je omejeno!

		1		4			5	5	
		5	1			2			
	3			2				2	
		5		2	2				3
	3					5			
	5				2	4			
		5							2
				1	4	2		5	

		5		1		5			5
		2	1					3	
	5				2				2
					5		5		3
	3							5	2
	2				2		4		
		5					5		3
	1			1	4		2		4

### Povezava do naloge


### Rešitev




### Ideja reševanja

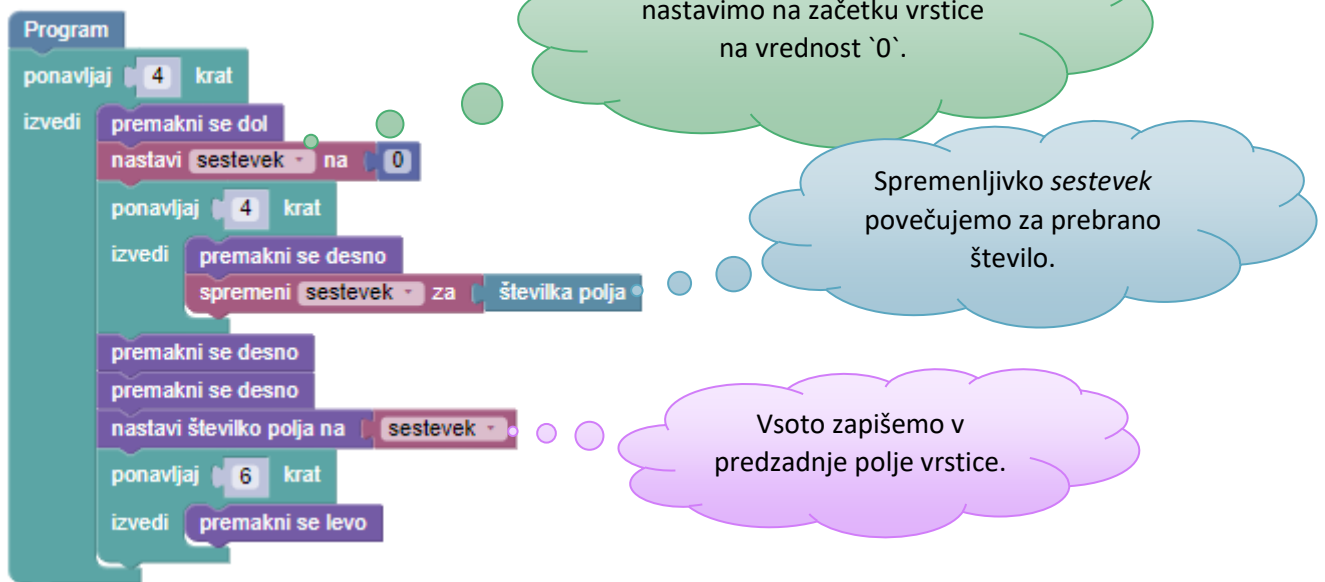
Z uporabo preproste zanke se Pišek premika po vrsticah, med tem pa s pogojnimi stavki preverja, ali je na polju število 5, in če je, število pobere. Zaradi omejitve delčkov mora Pišek pregledovati vrstice v obeh smereh.

## OŠ 7. – 9. r. NAPREDNI

							
	1	2	3	2		0	
	1	0	0	1		0	
	1	2	3	3		0	
	0	2	1	1		0	

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

## Rešitev



## Ideja reševanja

Najprej si dobro ogledamo oba testna primera. Hitro lahko opazimo, da ima vsak test 4 vrstice, v vsaki so 4 števila, ki jih mora Pišek sešteti, sledi prazno polje in nato polje, kamor mora vpisati vsoto. Pomembno je, da preden začne Pišek seštevati števila, vedno na začetku vrstice nastavimo vrednost spremenljivke *sestev* na `0`, nato pa to spremenljivko povečujemo za vrednost, ki je na polju. Če tega ne bi naredili, bi v naslednji vrstici prištevali števila k vsoti iz prejšnje vrstice, česar pa ne želimo. Na koncu vrstice nastavimo številko polja na vrednost, ki je shranjena v spremenljivki *sestev* in se vrnemo nazaj na začetek vrstice. Z uporabo zunanje zanke postopek ponovimo štirikrat.

## UČITELJ GROZNI

OŠ 7. – 9. r. NAPREDNI

Učitelj Grozni je res grozni učitelj. Vsi učenci se ga grozno bojijo, saj ocenjuje zelo grozno. A njegovo ocenjevanje ni povsem naključno, ali pač?

Namreč ... Učitelj Grozni pri svojem ocenjevanju uporablja program, ki naključno določi 3 stvari in jih izpiše. Prvič: Ali bo učenec ocenjen »ustno« ali »pisno«. Drugič: Ali bo učitelj Grozni med ocenjevanjem gledal »zelo grdo« ali »manj grdo«. In tretjič: Katero oceno bo dobil učenec: 1, 2, 3, ali 4 (ocene 5 učitelj Grozni namreč ne daje). Dopolni program tako, da bo le-ta delal pravilno.

Spodaj so primeri možnih izpisov programa.

Output:  
pisno  
manj grd  
2

Output:  
pisno  
zelo grd  
1

Output:  
ustno  
manj grd  
4

Input:	Output:

[Povezava do naloge](#)

## Rešitev



Naključno generiranje števil,  
na podlagi katerih se učitelj  
Grozni odloči za tip  
ocenjevanja.

Naključno generiranje števil,  
na podlagi katerih se učitelj  
Grozni odloči za pogled med  
ocenjevanjem.

Naključno  
generiranje ocene.

## Ideja reševanja

Pri nalogi je potrebno na podlagi prebranega problema, ugotoviti, da moramo v danem programu »sami« določiti »vhodne« podatke. Vhodni podatki pa so števila, ki jih je potrebno »naključno izbrati«. Da bi posamezni vhodni podatek vselej naključno izbrali (torej ustno ali pisno; zelo grd ali manj grd; itn.) je potrebno poznati delček za generiranje naključnega števila, ki ga najdemo pod delčki za matematiko. Ta delček je potrebno uporabiti pri vseh treh »stvareh« (ki jih prof. Grozni naključno izbira pri svojem ocenjevanju).

Ko generiramo posamezno naključno število (za posamezno stvar), lahko vrednost, ki je bila generirana shranimo v spremenljivko. To naredimo v primeru določanja ocene, ko naključno generirano število med 1 in 4, že predstavlja oceno.

Delček za izbor naključnega števila pa v prvih dveh primerih (tip in pogled) lahko kar vstavimo v pogoj pogojnega stavka. V teh dveh primerih pa generiramo le naključno število med 1 in 2, ker imamo le dve možnosti tipa ocene in pogleda učitelja.

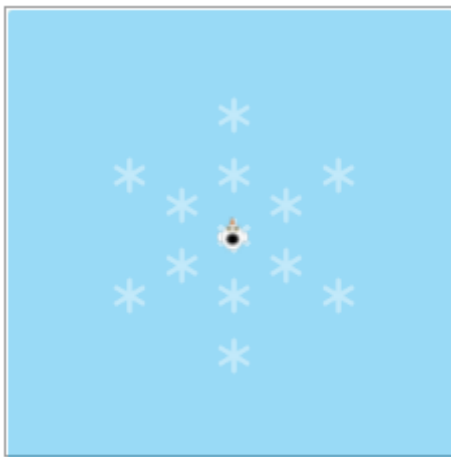
Ko smo pridobili vse tri naključno izbrane stvari (tip, pogled in oceno), jih na koncu, v pravilnem vrstnem redu, z delčkom »izpiši«, tudi izpišemo. Možne so tudi nekoliko drugačne rešitve.



## SNEŽAK RIŠE SNEŽINKE

OŠ 7. – 9. r. NAPREDNI

Snežak ima zelo rad sneg. Najraje pa na lep sončen dan na poledenelem jezeru za hišo riše snežinke. Vedno riše povsem enake snežinke, ki imajo krak dolžine 10. Tokrat pa bo narisal kar 13 takšnih snežink, razdalja med snežinkami pa je 40 enot. Napiši program, ki bo na led narisal takšno sliko, kot jo vidiš spodaj, pri tem pa ustvari funkcijo za risanje ene snežinke.



### Povezava do naloge

### Rešitev



## Ideja reševanja

Najprej si dobro ogledamo sliko. Če sliko pogledamo »bolj od daleč«, vidimo eno veliko snežinko, ki pa jo v resnici sestavlja 13 povsem enakih snežink. Da si olajšamo delo, uporabimo funkcijo za risanje ene snežinke, ki je že pripravljena na delovni površini. Pri risanju ene snežinke, se Snežko premakne naprej za 10 korakov, nato mora dvigniti pisalo in se premakniti nazaj na izhodiščno točko. Ker Snežko riše 6-krako snežinko, se mora obrniti za 60 stopinj v levo ali v desno. To pa ponavlja 6-krat, torej za vsak krak posebej, za kar uporabimo preprosto zanko.

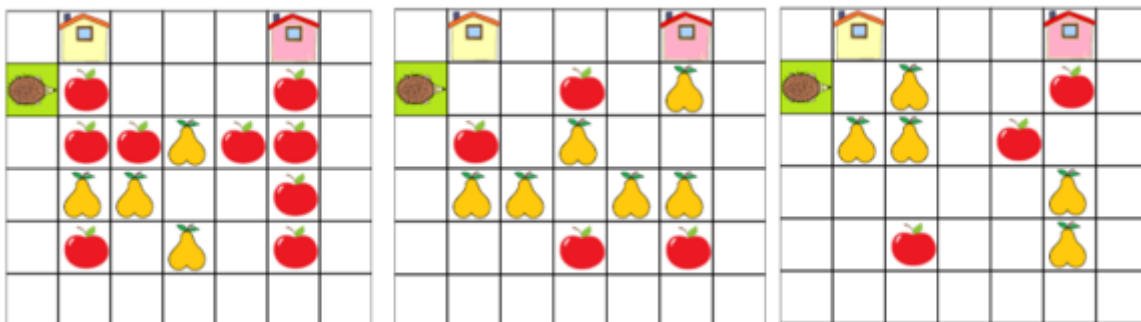
Ko imamo funkcijo za risanje ene snežinke pripravljeno, se lahko lotimo glavnega programa. Napisani program nam že nariše sredinsko snežinko, mi pa moramo dopisati še program za ostale. Opazimo lahko vzorec, in sicer so vse snežinke razporejene v obliki 6-krake snežinke, z dolžino kraka 40 oz. 80 korakov. Snežko se torej premakne naprej za 40 korakov in nariše snežinko, kar naredimo s klicem funkcije `snezinka`. To ponovi dvakrat, nato pa se mora vrniti nazaj za 80 korakov. Ker so snežinke razporejene v obliki 6-krake snežinke, se tudi sedaj Snežko obrne za 60 stopinj v levo ali pa v desno, in postopek ponavlja za vsak krak velike zunanje snežinke, torej natanko 6-krat.

## JEŽEK POSPRAVLJA

OŠ 7. – 9. r. NAPREDNI

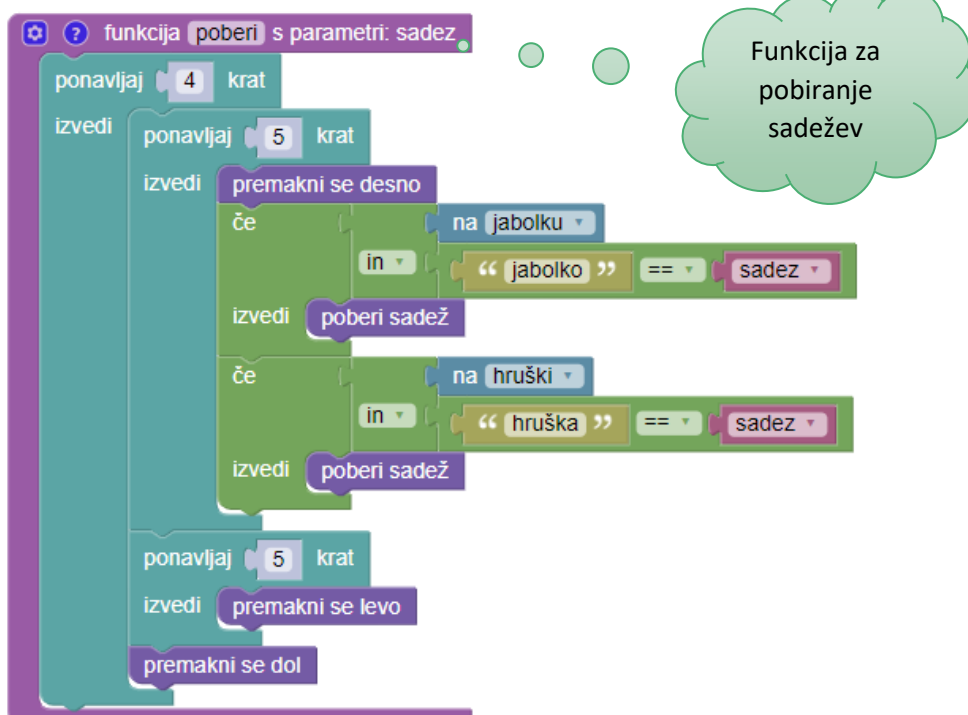
Ježek se jeseni pripravlja na zimo, a lahko najame le eno skladišče, rumeno za hruške, ali pa rdeče za jabolka. Ker želi skladišče čim bolj izkoristiti, bo najprej pregledal cel travnik, da ugotovi, katerih plodov je več. Če bo hrušk več kot jabolk, jih bo pobral in shranil v rumeno skladišče, sicer pa bo pobral jabolka in jih shranil v rdeče skladišče. Število jabolk ni nikoli enako številu hrušk.

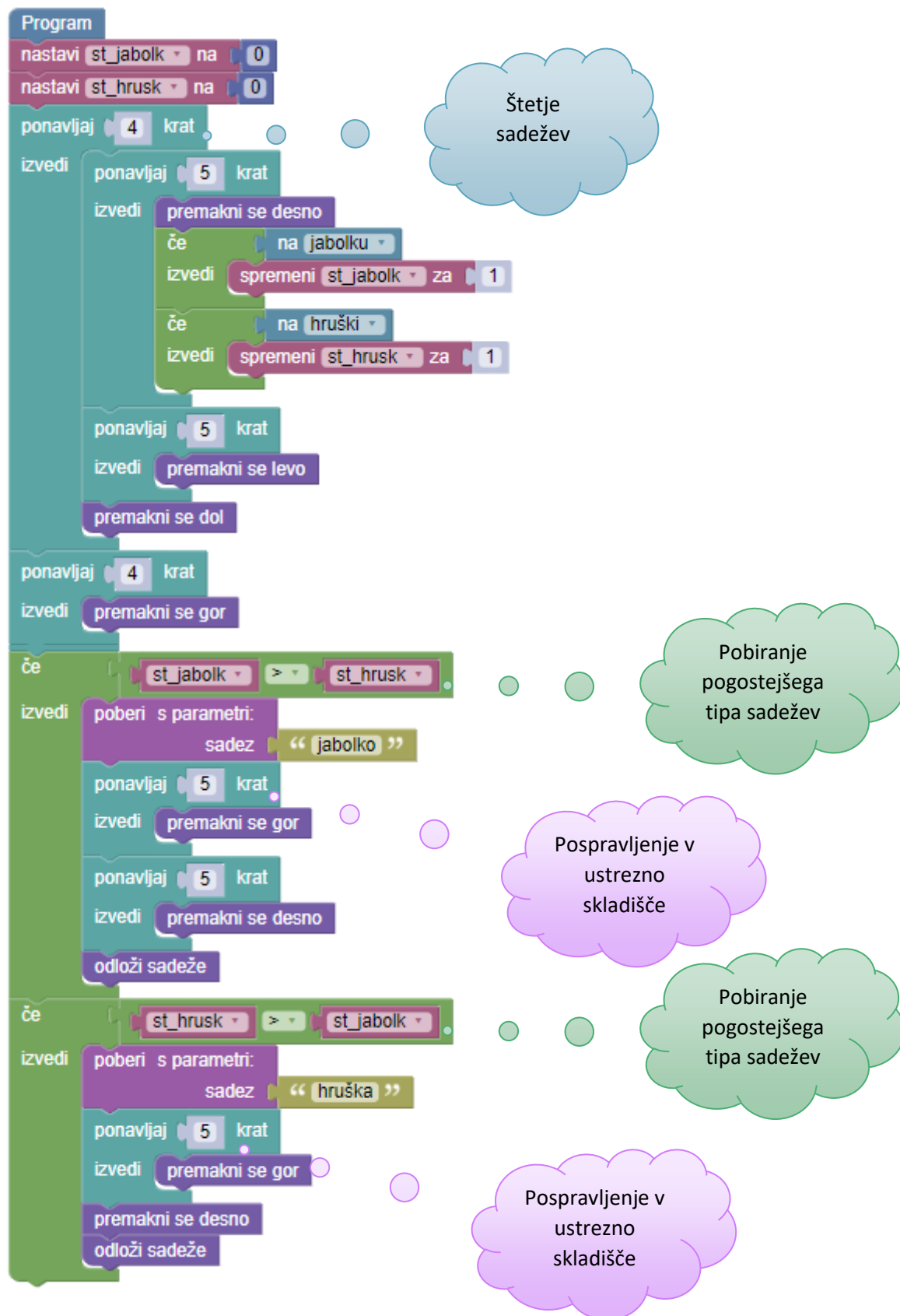
Napiši kodo, s katero bo Ježek ustrezno preštel sadeže in se vrnil na zeleno polje. Nato pokliči funkcijo "poberi", ki pobere sadeže, ki jih damo v argumentu. Na koncu pa naj Ježek ustrezne sadeže odnese in pospravi v njihovo skladišče.



### Povezava do naloge

## Rešitev





## Ideja reševanja

Naloga je zelo kompleksna in jo najlažje rešimo, če jo razbijemo na manjše podprobleme. Ugotovimo lahko, da bo naša koda sestavljena iz treh delov: 1) štetje sadežev, 2) pobiranje pogostejšega tipa sadežev, 3) pospravljanje v ustrezno skladišče.

Najprej se mora torej ježek premakniti po naši mreži in prešteti vse sadeže. Dobro si ogledamo vse tri testne primere in ugotovimo, da mora ježek pregledati le manjšo mrežo v velikosti 5x4. Sprehod po tem delu mreže naredimo z uporabo dvojne zanke. Med sprehodom pa mora ježek že preštovati sadeže, prešteto pa si mora tudi zapomniti. Potrebujemo torej dve novi spremenljivki; mi smo ju poimenovali *st\_jabolk* in *st\_hrusk*, lahko bi ju seveda tudi drugače. Na začetku moramo nastaviti vrednost obeh spremenljivk na 0, nato pa med sprehodom po mreži uporabimo pogojni stavek s senzorjem in povečujemo vrednost ustrezne spremenljivke za +1.

Drugi podproblem s pobiranjem sadežev je delno že rešen, saj imamo na delovni površini že pripravljeno funkcijo »poberi s parametrom sadež«. Če si dobro ogledamo funkcijo, ugotovimo, da mora biti ježek na zelenem polju preden pokličemo funkcijo, da bo le-ta tudi pravilno delovala. Funkcija ima dva parametra: jabolko in hruška; če še enkrat dobro preberemo besedilo, vidimo, da moramo klicati funkcijo s parametrom jabolko takrat, ko bo ježek preštel, da je na travniku več jabolk, oz. ko bo vrednost spremenljivke *st\_jabolk* večja od vrednosti spremenljivke *st\_hrusk* in obratno v nasprotnem primeru.

Zadnji podproblem pa je pospravljanje v ustrezno skladišče. Premike do rdečega skladišča s preprostimi zankami vstavimo kar v pogojni stavek, kjer je število jabolk večje od števila hrušk in podobno tudi za rumeno skladišče. Na koncu ne smemo pozabiti, da mora ježek sadeže v obeh primerih tudi odložiti.

Možne so tudi nekoliko drugačne rešitve, npr. namesto dveh pogojnih stavkov, lahko uporabimo tudi en pogojni stavek tipa `če-sicer`.

## KEMIK ŽIGA

## SŠ ZAČETNIKI

---

Kemik Žiga izvaja kemijski poskus. V priročniku je našel seznam s količinami vseh potrebnih kemikalij v gramih (g). Šef pa mu je sporočil, da so v priročniku napake. Zato mora za uspešno izvedbo povečati maso vsake kemikalije za točno določen odstotek (%).

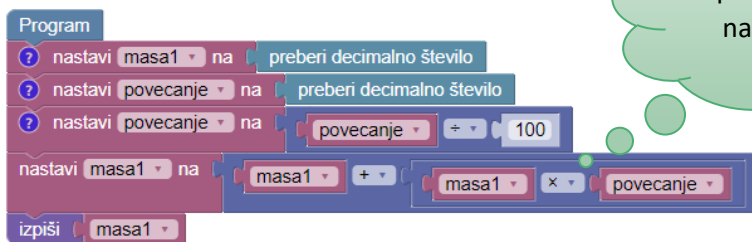
Žiga je v beležko (Vhod:) zapisal najprej maso v gramih iz priročnika, nato pa še povečanje mase v odstotkih (%). Nato je napisal program, ki naj bi izračunal pravilno maso vseh kemikalij. Program pa ne deluje pravilno. Pomagaj mu.

Input:	Output:
1.99	
28	

[Povezava do naloge](#)

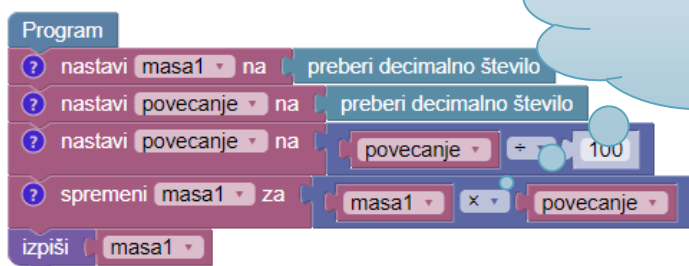
## Rešitev

### 1. rešitev



Vrednost  
spremenljivke `masa1`  
na novo nastavimo.

### 2. rešitev



Vrednost spremenljivke  
`masa1` zgolj spremenimo za  
izbrani odstotek.

## Ideja reševanja

V nalogi je potrebno odkriti napako v že predlaganih delčkih. Po pregledu delčkov opazimo, da je napaka v peti vrstici, saj je napačno uporabljen delček `spremeni`.











V 1. rešitvi, smo zamenjali le »zunanjí delček« iz `spremeni` v `nastavi` in tako na »novo« nastavili spremenljivko `masa1`. Pri 2. rešitvi, pa smo trenutno vrednost spremenljivke `masa1` le povečali oz. spremenili za izbrano maso v odstotkih.

## BOŽIČKOVA OKRASITEV

SŠ ZAČETNIKI

Božiček je v zaključku preteklega leta prosil Piška za pomoč pri okrasitvi mesta, vendar se takrat ni najbolje izteklo. Zato se bo letos bolje pripravil; in čeprav je do decembra še kar veliko časa, že ukrepa!

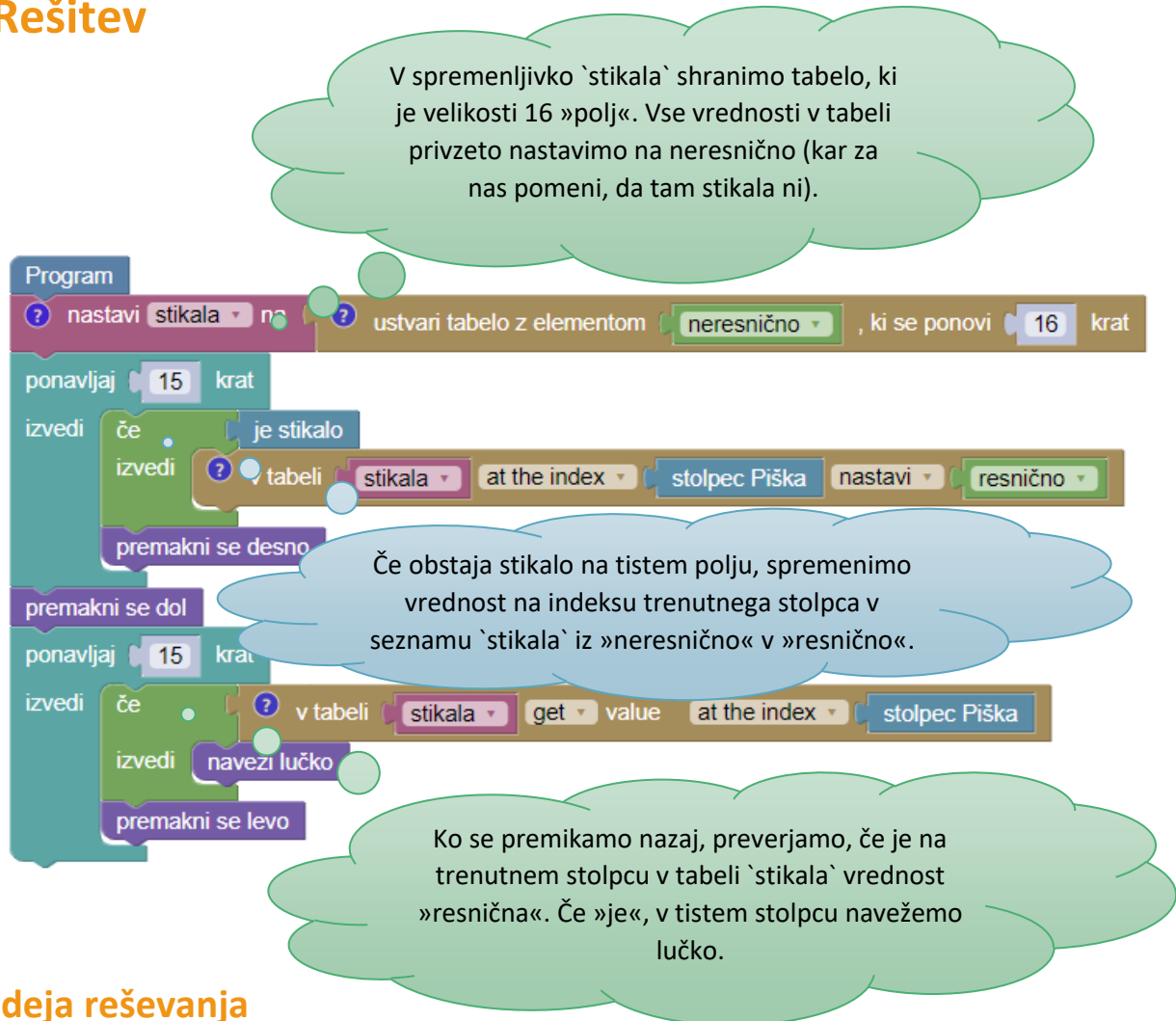
Za Piška bo pripravil točna navodila. Praznično vzdušje bo pričaral z božičnimi lučkami, ki jih je potrebno navezati na polja pod stikali. Zaradi »tehničnih težav« pa Pišek ob navezovanju lučk ne sme zapustiti vrstice z lučkami. Zato si mora najprej zapomniti, katera polja v prvi vrstici imajo stikala in šele nato navezovati lučke.

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

[Povezava do naloge](#)



## Rešitev



## Ideja reševanja

Da si bo Pišek zapomnil, kje se nahajajo stikala smo ustvarili tabelo (z imenom `stikala`), ki je velikosti oz. dolžine 16 (polj). V to ustvarjeno tabelo bomo beležili na katerih poljih so stikala. Ob ustvarjanju tabele, pa je potrebno nastaviti tudi privzete vrednosti. Dobra ideja je, če »privzete vrednosti« v tabeli nastavimo na »neresnično«, kar bo v našem primeru pomenilo, da stikala v tistem stolpcu ni. V primeru, da se bo kasneje izkazalo (ko se bomo s Piškom premikali iz leve proti desni), da je stikalo v določenem stolpcu »prisotno«, pa bomo vrednost na tistem indeksu oz. stolpcu spremenili iz »neresnično« v »resnično«.

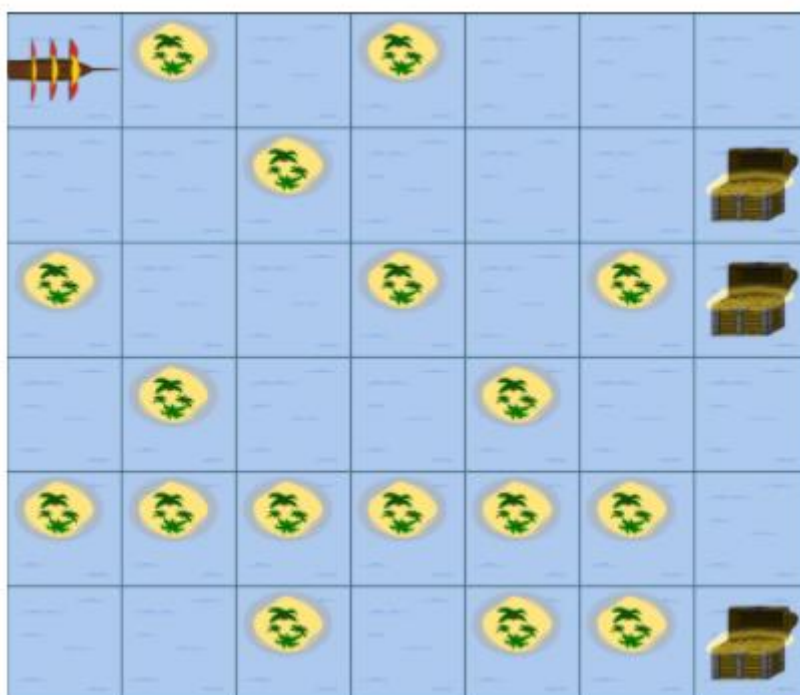
Potem ko smo »ustvarili« tabelo, je potrebno sprogramirati, da se bo Pišek premikal iz leve proti desni. Prav tako pa je potrebno pred vsakim premikom v desno, preveriti, če je na Piškovem trenutnem mestu oz. stolpcu stikalo. Kot smo že dejali, bomo v primeru, da je stikalo v Piškovem trenutnem stolpcu »prisotno«, vrednost v tabeli, ki je na indeksu trenutnega Piškovega stolpca spremenili iz »neresnično« v »resnično«. Vse to je potrebno ponoviti 15 krat.

Ko smo prišli na konec prve vrstice, pa se je potrebno premakniti navzdol in se nato pričeti premikati nazaj proti stolpcu ena. Ob vsakem premiku, pa tokrat preverjamo, če je vrednost v tabeli, na trenutnem indeksu (torej stolpcu Piška) »resnična«, kar pomeni, da se nahajamo v stolpcu, kjer je v zgornji vrstici stikalo in je tako potrebno navezati lučko. Tudi slednje ponovimo 15 krat.

## ZAKLAD CEKINOV

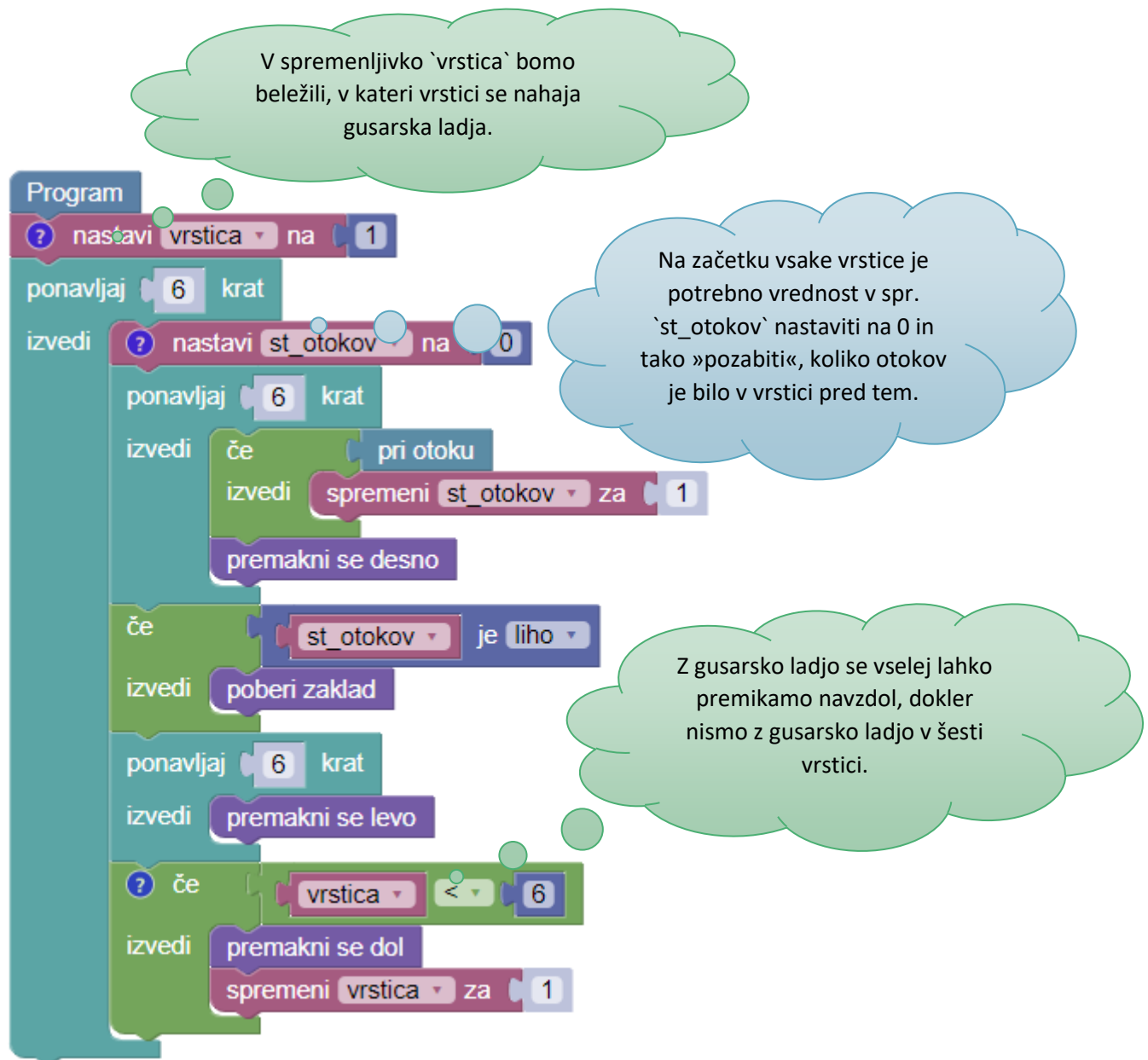
SŠ ZAČETNIKI

Gusarji so našli več zemljevidov, ki jih pripeljejo do zakladov. Zakladi se nahajajo v tistih vrsticah, ki imajo liho število otokov. Zapiši program, ki za oba testna primera oz. zemljevida, gusarje, vodi po vrsticah in šteje otoke. V kolikor pa gusarji v posamezni vrstici naštejejo liho število otokov, pa naj na koncu tudi poberejo zaklad!



[Povezava do naloge](#)

## Rešitev



## Ideja reševanja

Pred pričetkom reševanja premislimo naslednje. Z gusarsko ladjo se je potrebno premikati skozi šest vrstic. V vsaki vrstici, se je potrebno z ladjo premakniti šestkrat iz leve proti desni in se nato še vrniti nazaj na izhodišče (na prvi kvadrateg). Šele nato pa se premaknemo v drugo vrstico. Vse to je potrebno storiti šestkrat, razen v zadnji vrstici, ko je potrebno biti pazljiv, da se z ladjo ne premaknemo navzdol, saj bomo v nasprotnem primeru zapustili mrežo. Zato je potrebno (vselej) vedeti, v kateri vrstici se nahaja gusarska ladja. V ta namen smo ustvarili spremenljivko `vrstica`, v katero bomo beležili, koliko vrstic je ladja gusarjev že prepotovala.

Kot smo že dejali, se z ladjo premikamo skozi šest vrstic. V vsaki vrstici pa je potrebno prešteti število otokov. Da bi prešteli število otokov v posamezni vrstici, smo v ta namen ustvarili spremenljivko `st\_otokov`, ki jo povečamo za »+1«, kadar smo z gusarsko ladjo pri otoku. Na koncu, v šestem kvadratu, pa preverimo, če je število v spremenljivki `st\_otokov` liho, in če »je liho«, v šestem kvadratu poberemo zaklad. Nato pa se z gusarsko ladjo vrnemo nazaj na prvi kvadrateg v vrstici in gremo eno vrstico navzdol. Pri prehodu v novo vrstico pa je potrebno biti pozoren na dve stvari. Povečati je potrebno vrednost v spremenljivki `vrstica` za »+1« in nastaviti vrednost v spremenljivki `st\_otokov` nazaj na »0«, saj je potrebno v novi vrstici šteti otoke »od začetka«.

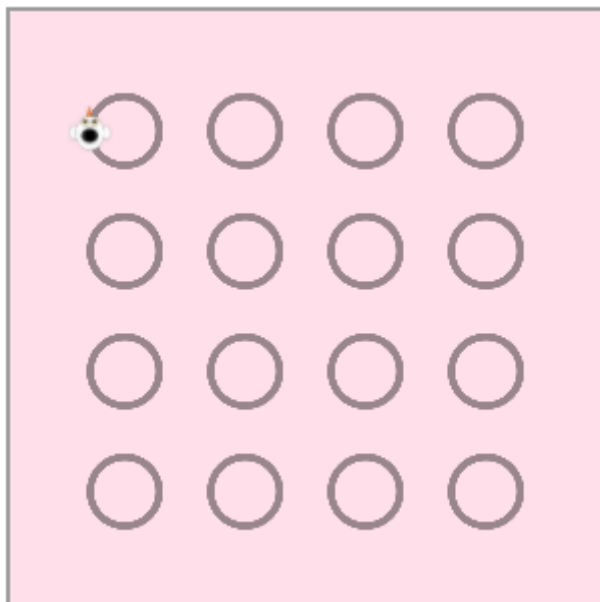
## SNEŽKO SNEŽAK IGRA TRI V VRSTO

SŠ ZAČETNIKI

Snežko Snežak bi rad s prijatelji igral tri v vrsto. Vendar, ker je notri pretoplo, se lahko igrajo le zunaj, na snegu. Odločil se je, da bo na tla narisal igralno polje in sicer, v štiri vrstice, po štiri krogce. Pomagaj mu napisati program, ki ga bo vodil do pravilne talne risbe.

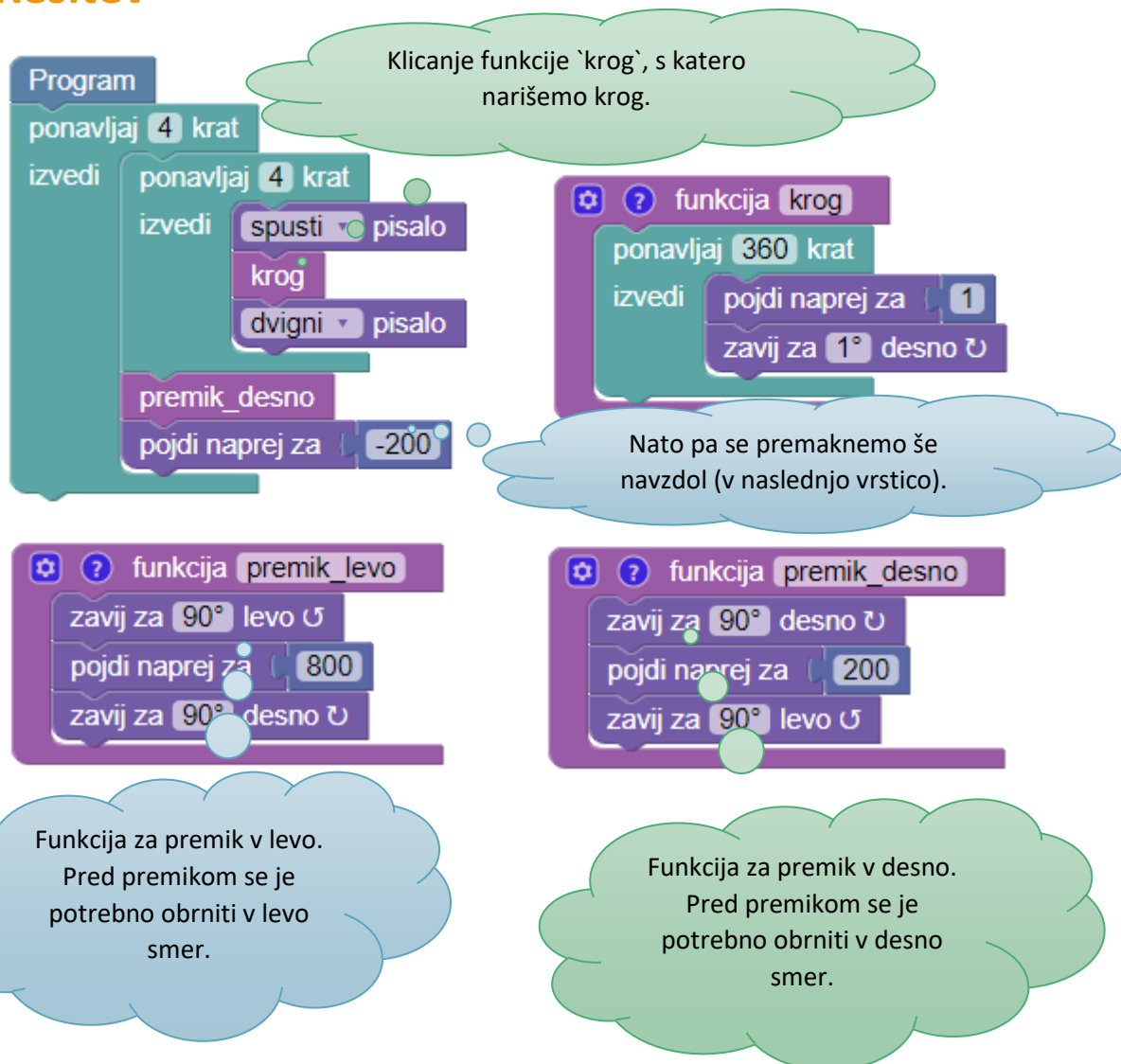
*Namig 1.* Oddaljenost dveh krogov je 200 enot.

*Namig 2.* "Krog" narišeš tako, kakor, da bi narisal pravilni 360-kotnik! Za hitrejšo animacijo klikni spodaj na četrti modri gumb.



[Povezava do naloge](#)

## Rešitev



## Ideja reševanja

Pri tej nalogi imamo že podano funkcijo za `krog`, ki jo je potrebno nekoliko predelati, da bomo s podano funkcijo narisali krog. Kot je že zapisano v namigu lahko »krogu podoben lik« narišemo tako, da narišemo 360-kotnik, ki ga narišemo tako, da se 360-krat obrnemo za 1 stopinjo (v desno), ob vsakem zasuku za 1 stopinjo pa se pomaknemo naprej za 1 enoto (seveda to velja le v našem primeru in ne nasplošno).

Ko smo napisal funkcijo za krog, pa je potrebno le še premisliti, kako se premikati v desno po vrstici in nato po vrsticah navzdol. V naši rešitvi smo v ta namen definirali dve dodatni funkciji.

S funkcijo `premik \_desno` smo sprogramirali, da se Snežko premakne v desno. Pri tem pa je potrebno biti pozoren, da pred »premikom v desno« dvigne pisalo in se obrne za 90 stopinj v desno. In se šele nato izvede sam »premik« (za 200 enot). Podobno velja za funkcijo `premik \_levo`, le da je tam premik nekoliko daljši (800 enot).

Vse omenjene funkcije pa nato uporabimo v glavnem programu. Snežko nariše »kroge« tako, da v vsaki vrstici nariše štiri kroge in se nato pomakne nazaj na izhodišče trenutne vrstice. Nato pa gre v naslednjo vrstico.

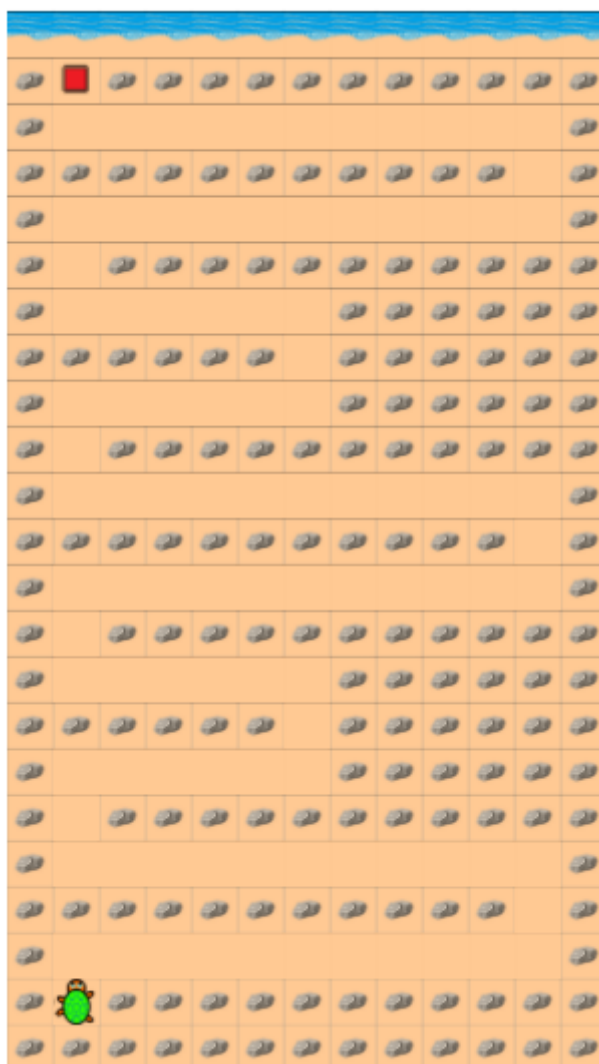
Seveda je to le ena izmed možnih idej. Npr. lahko bi se Snežko premikal tudi navpično navzdol in nato desno po stolpcih. Prav tako pa obstaja tudi rešitev brez funkcij, ki pa naj jo reševalec premisli sam.



## ŽELVICA KAJA

SŠ ZAČETNIKI

Morska želvica Kaja se je ravnokar izlegla iz jajčeca. Njena naloga je, da čim hitreje pride v varno okolje morja (rdeči kvadratik). Pomagaj ji najti pot; pri tem pa pazi, da ne presežeš dovoljenega števila delčkov. Prav tako pa pri programiranju uporabi že podano funkcijo, ki pa ima nekaj napakic!

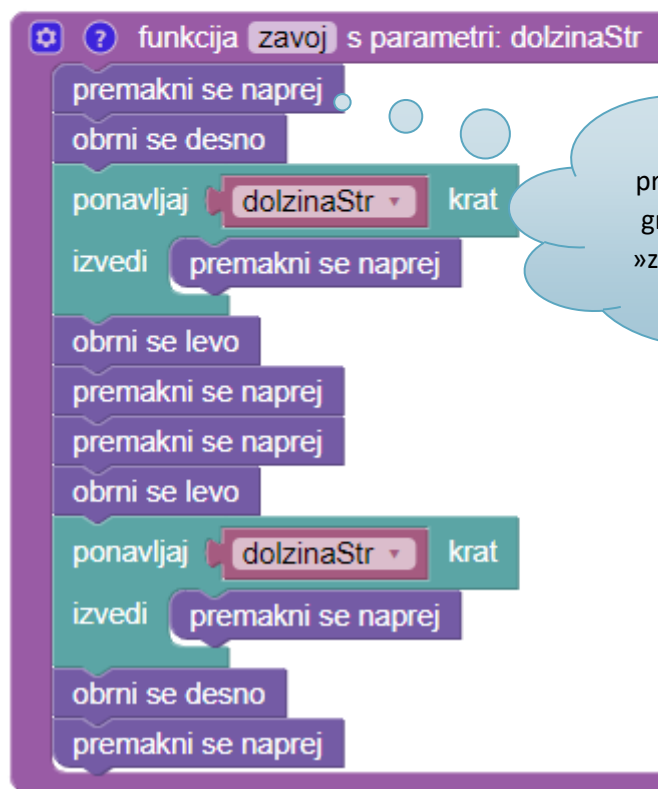


[Povezava do naloge](#)

## Rešitev



Funkcija `zavoj` s parametrom `dolzinaStr`, s katerim določimo, koliko korakov v desno in levo mora želvica iti.



Funkcija, s katero želvico premikamo po mreži. Na začetku gre vedno korak naprej, nato pa »za vijuga« desno in nato še levo.

## Ideja reševanja

Pri tej nalogi je potrebno popraviti funkcijo `zavoj` in nato še sprogramirati glavni program, v katerem uporabimo popravljeno funkcijo `zavoj`.

Če pred pričetkom reševanja pogledamo mrežo, na kateri se nahaja morska želvica, opazimo, da se od njenega gnezda in vse do morja pojavljata dva različna vzorca, po katerih se želvica premika. Enkrat se želvica desno in levo premika za »10 korakov«, drugič pa se želvica premika desno in levo za »5 korakov«. Torej

je potrebno funkcijo ``zavoj`` prilagoditi tako, da se bo želvica vselej premikala glede na trenutni vzorec. Prav tako, pa lahko pri funkciji ``zavoj`` opazimo, da leta sprejme parameter ``dolzinaStr``, ki bi ga lahko uporabili za določitev števila korakov v levo in desno.

Rešitev naloge se po našem premisleku ne zdi tako težka. Sprva je potrebno v obeh zankah (ki sta v funkciji) uporabiti parameter ``dolzinaStr`` in tako določiti, da se bo želvica premaknila le za ``dolzinaStr`` korakov (v desno in nato še v levo) in ne vselej za 10.

Poleg tega pa v funkciji opazimo, da je želvica malce »nehote« zamenjala smeri »levo« in »desno«. Torej je potrebno vsak delček ``obrni se levo`` zamenjati z delčkom ``obrni se desno``; in podobno, vsak delček ``obrni se desno`` zamenjati z delčkom ``obrni se levo``.

V nadaljevanju pa je potrebno funkcijo ``zavoj`` uporabiti v glavnem programu, kar pa bralcu ne bi smelo predstavljati (večjega) problema.

## PROFESOR GROZNI

SŠ NAPREDNI

Profesor Grozni je res grozni profesor in vsi dijaki se ga grozno bojijo, saj ocenjuje zelo grozno! A njegovo ocenjevanje ni povsem naključno, ali pač?

Namreč ... Profesor Grozni uporablja pri svojem ocenjevanju program, ki naključno »generira« in izpiše naslednje tri stvari. Prvič. Ali bo dijak ocenjen »ustno« ali »pisno«. Drugič. Ali bo (profesor Grozni) med ocenjevanjem gledal »zelo grdo« ali »manj grdo«. In tretjič. Katero oceno bo dijak dobil: »nezadostno«, »zadostno«, »dobro«, ali »prav dobro« (ocene »odlično« pa profesor Grozni ne daje!). Dopolni (spremeni) program tako, da bo le-ta deloval pravilno.

Spodaj so primeri možnih izpisov programa

Output:  
ustno  
manj grd  
zadostno

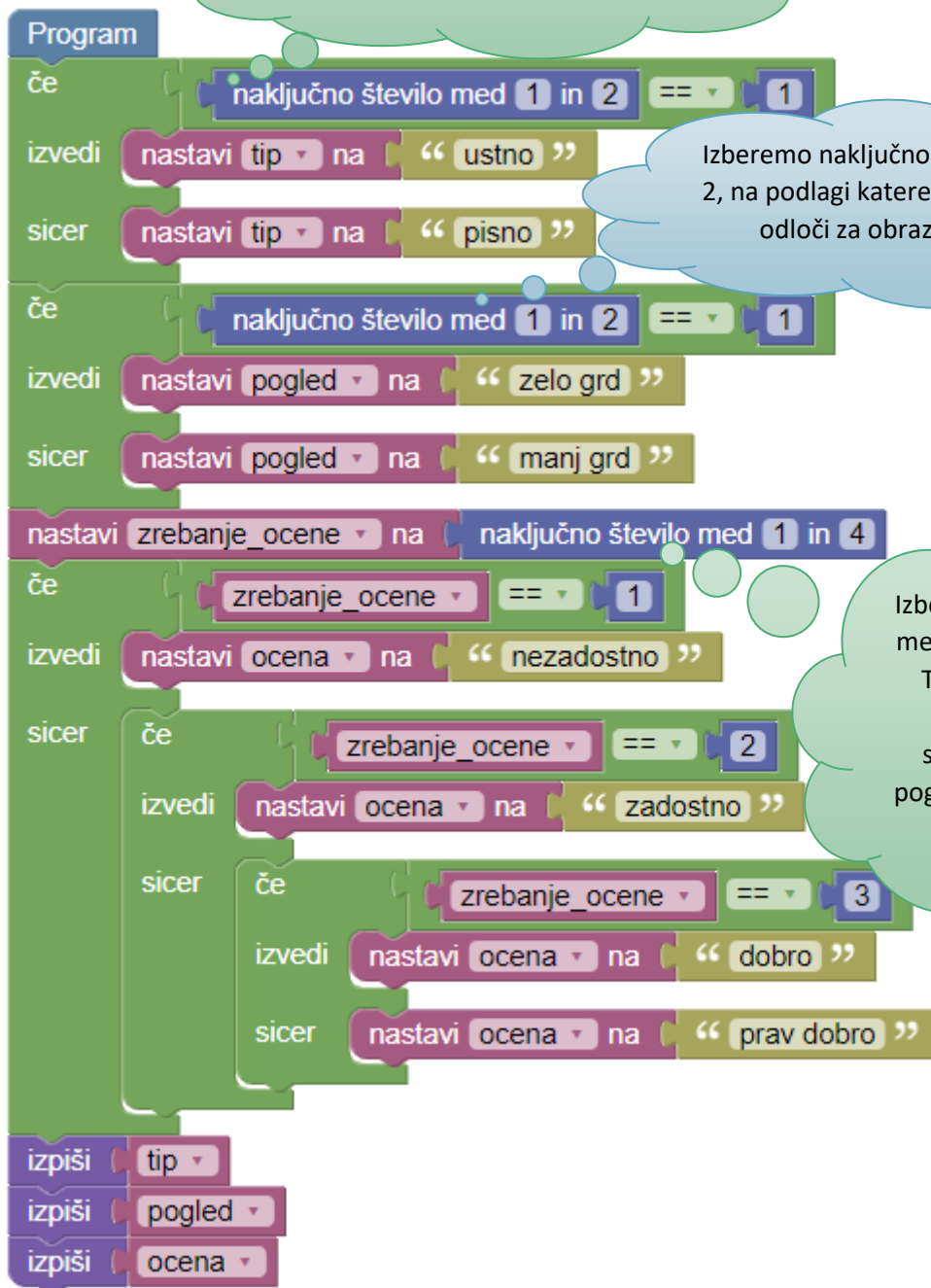
Output:  
ustno  
zelo grd  
dobro

Output:  
pisno  
manj grd  
nezadostno

Input:	Output:

[Povezava do naloge](#)

## Rešitev



## Ideja reševanja

Pri nalogi je potrebno, sprva, na podlagi prebranega problema, ugotoviti, da moramo v danem programu »sami« določiti »vhodne« podatke. Vhodni podatki pa so števila, ki jih je potrebno »naključno izbrati«. Da bi posamezni vhodni podatek vselej naključno izbrali (torej ustno ali pisno; zelo grd ali manj grd; itn.) je potrebno poznati delček za generiranje naključnega števila, ki ga najdemo pod

delčki za matematiko. Ta delček je potrebno uporabiti pri vseh treh »stvareh« (ki jih prof. Grozni naključno izbira pri svojem ocenjevanju).

Ko generiramo posamezno naključno število (za posamezno stvar), lahko vrednost, ki je bila generirana shranimo v spremenljivko. Vendar to niti ni vselej potrebno, saj lahko delček za izbor naključnega števila, za prvi dve stvari (torej za tip in pogled), kar direktno vstavimo v prva dva pogojnika (saj tu izbiramo zgolj med dvema alternativama).

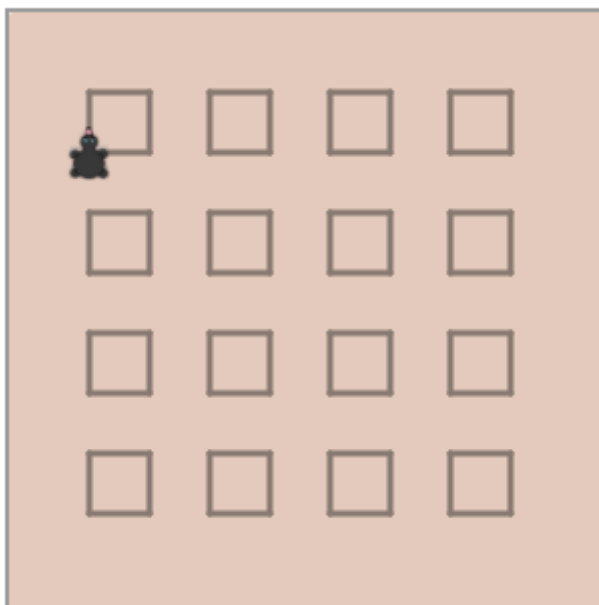
Za izbiro ocene pa tega ne smemo storiti, saj je potrebno naključno število med 1 in 4 najprej izbrati in nato shraniti v ustrezno spremenljivko. Vrednost, ki je shranjena v tej spremenljivki, pa nato uporabimo v »pogojnikih«, s katerimi določimo oceno. S tem zagotovimo, da je ta (naključno izbrana) vrednost pri vseh treh pogojnikih (za oceno) vselej enaka (in se ne generira sproti). Ko smo pridobili vse tri naključno izbrane stvari (tip, pogled in oceno), jih na koncu, v pravilnem vrstnem redu, z delčkom `izpiši`, tudi izpišemo.

## KRTOVA GLEDALIŠKA PREDSTAVA

SŠ NAPREDNI

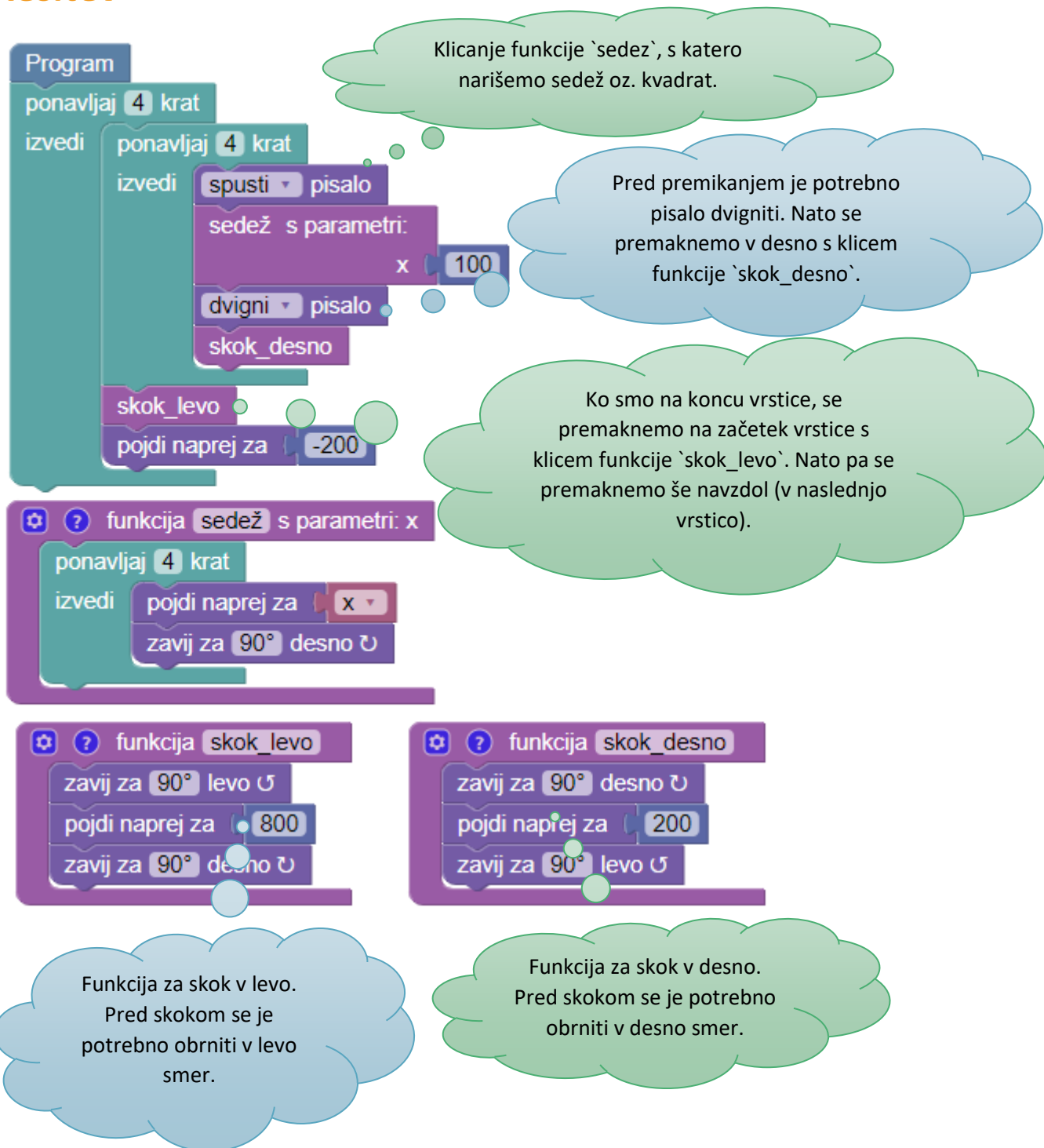
Krt je znan po tem, da vedno skrbi za kulturno dogajanje na domačem travniku. Tudi letos bo poskrbel za gledališko predstavo. Ker pa v letošnjem letu veljajo hudi epidemiološki ukrepi, mora skrbno načrtovati postavitev sedežnega reda za vse gledalce.

Za namen svoje predstave bo pripravil šestnajst prostih mest, ki jih bo razporedil v vrsto po štiri. Med sedeži pa mora poskrbeti tudi za dovolj prostora. Pomagaj krtu napisati program, ki bo narisal sedežni red, kot ga vidiš na spodnji sliki.



[Povezava do naloge](#)

## Rešitev





## Ideja reševanja

Pri nalogi imamo v naprej podano funkcijo ``sedez``, ki sprejme parameter ``x``. Podano funkcijo je potrebno spremeniti, da bomo z njo narisali kvadrat (oz. sedež) z dolžino stranice ``x``.

Z malo eksperimentiranja ugotovimo, da mora stranica kvadrata meriti 100 enot. O funkciji za kvadrat pa tudi ni veliko za »razglablјati«. Da narišemo kvadrat, je potrebno, da se zanka izvede štirikrat, ob vsaki »iteraciji«, pa se premaknemo za parameter ``x`` enot naprej in se nato še obrnemo za 90 stopinj v ustrezno smer.

Ko pa smo napisal funkcijo za ``sedez``, pa je potrebno le še premisliti, kako se premikati desno po vrstici in nato po vrsticah navzdol. V naši rešitvi smo v ta namen definirali dve dodatni funkciji.

S funkcijo ``skok_v_desno`` smo sprogramirali, da krt skoči v desno za 200 enot. Podobno velja za funkcijo ``skok_v_levo``, le da je pri tej funkciji skok krta bistveno daljši (800 enot). Pri obeh skokih pa je potrebno biti tudi pozoren, da se pred vsakim skokom obrnemo v pravo smer.

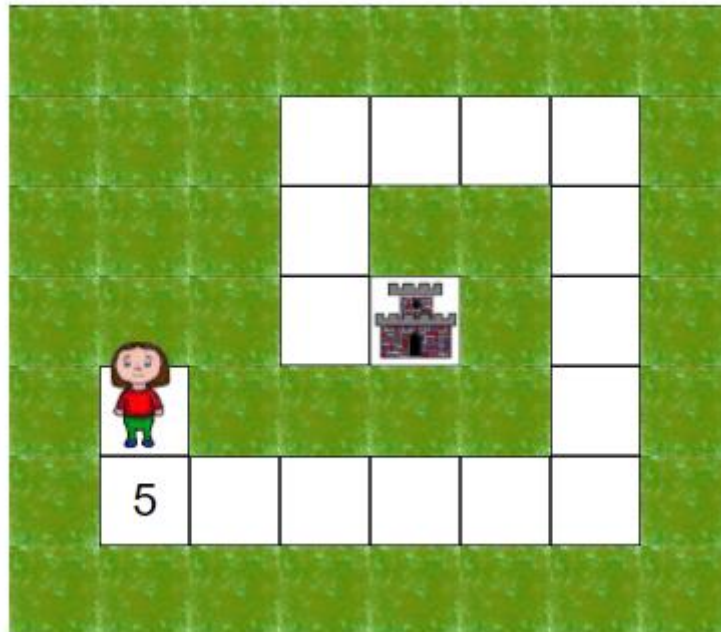
Vse omenjene funkcije nato uporabimo v glavnem programu. Pri tem je potrebno paziti, da imamo "spuščeno pisalo" le takrat, kadar želimo narisati sedež. V vseh preostalih primerih pa naj bo pisalo kar dvignjeno.

Seveda naj bralca spomnimo, da je to le ena izmed možnih idej. Torej obstaja tudi zadovoljiva rešitev brez funkcij, ki pa naj jo bralec premisli sam.

## MARTIN V LABIRINTU

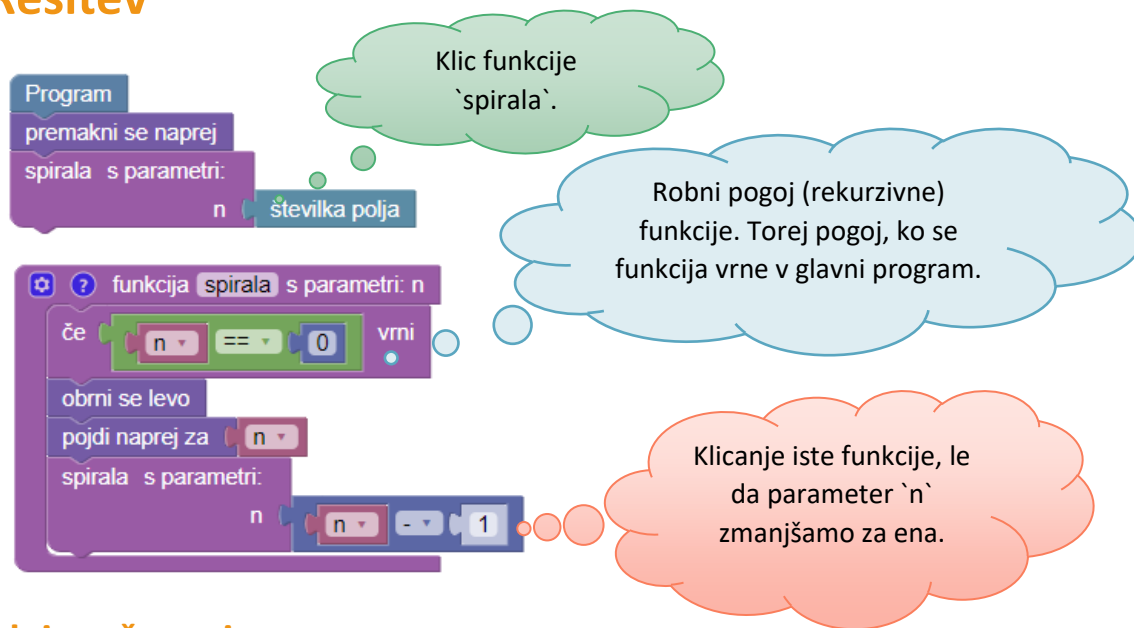
SŠ NAPREDNI

Pomagaj Martinu najti pot do gradu, kjer ga čaka njegova princesa. Pri tem je potrebno uporabiti funkcijo.



[Povezava do naloge](#)

## Rešitev



## Ideja reševanja

Pri tej nalogi je potrebno prepoznati vzorec, po katerem se giblje Martin. Opazimo, da se Martin v vseh testnih primerih giblje »v liku«, ki je podoben spirali. Ta se vije v levo smer, kar pomeni, da se Martin vselej obrača v levo smer.

Poleg tega pa pred Martinom opazimo število. To število (recimo, da je to število N) določa število korakov, ki jih mora Martin narediti pred svojim prvim zavojem.

Torej je naloga Martina ta, da prebere število N, ki je pred njim, se obrne v levo in nato naredi N korakov naprej. Nato se ponovno obrne v levo in naredi N - 1 korakov naprej; nato se ponovno obrne v levo in naredi še naslednjih N - 2 korakov naprej, itn. Vse to pa Martin ponavlja, dokler ne prispe do gradu oz. mu ni potrebno narediti nobenega korakov več in je N = 0.

Slednje lahko zapišemo v kratki (»rekurzivni«) funkciji. V našem primeru smo definirali funkcijo `spirala`, ki sprejme parameter `n`. »Robni« pogoj naše funkcije pa je, ko je parameter `n` enak 0. Torej to je pogoj, ko se funkcija »zaključi« in se vrnemo v glavni program. V kolikor pa robni pogoj ne velja, pa se Martin obrne v levo in gre naprej za `n` korakov. Nato pa sledi ponovni klic te iste funkcije (torej funkcije `spirala`), le da zmanjšamo parameter `n` za ena (rdeči oblaček). In kot že rečeno, se vse to ponavlja dokler parameter `n` ni enak 0.


## DEMOKRATIČNO ČIŠČENJE

SŠ NAPREDNI

V Piškovi družini imajo 15 prostorov, ki jih je potrebno počistiti. Kupili so robotski sesalnik, ki pa deluje na prav poseben način. In sicer: najprej se premika in bere oznake v 1., 2. in 3. vrstici; nato pa prične čistiti sobe, ki so 4. vrstici. Njegova pomanjkljivost pa je ta, da ne zna izmenično »brati« in »čistiti«. Torej, če zaide v sobe, ki jih je potrebno očistiti (četrta vrstica), ne zna priti nazaj v celice z oznakami (torej v prve tri vrstice).

»Zapovrh« pa se družinski člani ne strinjajo glede tega, katere sobe so potrebne čiščenja. Zato so se odločili, da bodo za vsako sobo glasovali. Družinski član (ki predstavlja številko vrstice) odda glas oz. oznako (temna siva celica), če meni, da je soba (oznaka stolpca) potrebna čiščenja.

Sprogramiraj robotski sesalnik tako, da bo le-ta upošteval demokratično načelo v družini; ta narekuje, da sesalnik počisti le tisto sobo, za katero sta vsaj dva družinska člana oddala glas, da je ta potrebna čiščenja.

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

[Povezava do naloge](#)

## Rešitev

Ustvarimo prazno tabelo, ki jo shranimo v spremenljivko `seznam\_sob`.

Z robotom se premikamo preko 15ih sob.

Če je polje označeno, povečamo število glasov za čiščenje te sobe.

Samo v zadnjih sobah se ne smemo premikati levo (oz. desno), saj sicer zapustimo mrežo.

Preverimo, če je bil oddan več kot en glas za čiščenje te sobe. Če je bil, je potrebno počisti to sobo.

Funkcija, ki poveča vrednost polja za »+1« v seznamu `seznam\_soba` na izbranem indeksu.

```

Program
nastavi seznam_sob na ustvari tabelo z elementom 0, ki se ponovi 16 krat

za soba od 1 do 15 s korakom 1
izvedi če polje je označeno
izvedi glasovanje s parametri: indeks soba
če soba < 15
izvedi premakni se desno
premakni se dol
za soba od 15 do 1 s korakom 1
izvedi če polje je označeno
izvedi glasovanje s parametri: indeks soba
če soba > 1
izvedi premakni se levo
premakni se dol
za soba od 1 do 15 s korakom 1
izvedi če polje je označeno
izvedi glasovanje s parametri: indeks soba
če soba < 15
izvedi premakni se desno
premakni se dol
za soba od 15 do 1 s korakom 1
izvedi če v tabeli seznam_sob get value at the index soba > 1
izvedi počisti sobo
če soba > 1
izvedi premakni se levo

funkcija glasovanje s parametri: indeks
v tabeli seznam_sob at the index indeks nastavi v tabeli seznam_sob get value at the index indeks + 1
  
```

## Ideja reševanja

Pri nalogi je potrebno pred reševanjem premisliti naslednje: (i.) kako se bomo s sesalnikom premikali po vrsticah (torej ali bomo šli po vrsticah »cikcak«; ali pa se bomo vselej, ko bomo prišli na konec vrstice vračali nazaj, na začetek vrstice); (ii.) kako bomo s sesalnikom beležil glasovanja v družini (npr. ali je potrebno ustvariti tri tabele, torej tabelo za vsakega člana posebej; ali lahko problem rešimo tako, da ustvarimo zgolj eno tabelo, v katero bi beležili vse glasove).

V rešitvi, ki je podana zgoraj se robot premika po vrsticah »cikcak« (oz. prebira vrednosti, ko gre v desno in v levo), prav tako pa v rešitvi uporabimo zgolj eno tabelo (ki smo jo poimenovali ``seznam_sob``), da beležimo glasove vseh članov družine.

Prav tako pa smo v naši rešitvi definirali funkcijo ``glasovanje``, s katero povečujemo vrednost v tabeli (``seznam_sob``) na izbranem indeksu za »+1«. Seveda pa se ta funkcija kliče le takrat, ko je kvadrateg, na katerem se nahaja sesalec, pobarvan.

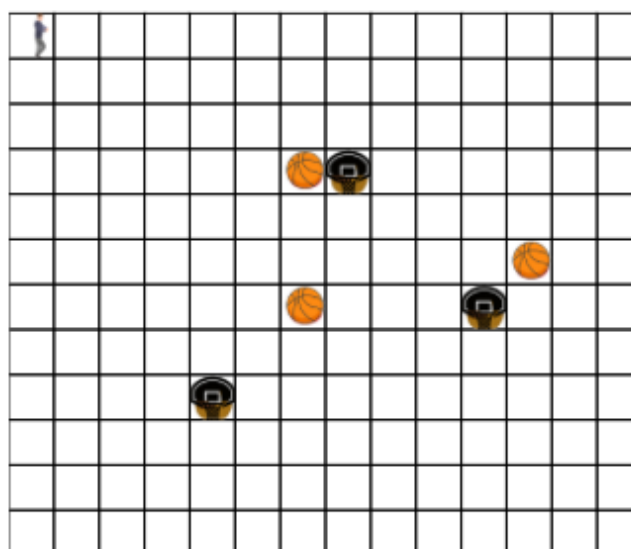
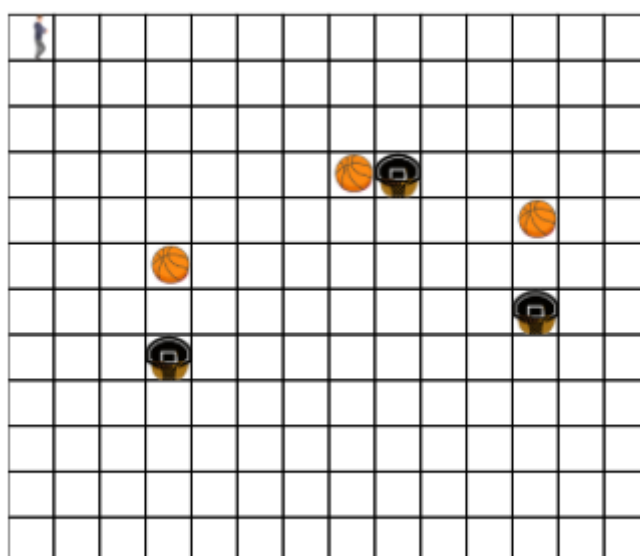
Ob vsem tem, pa velja opozoriti, da naj bo reševalec, v zgornji rešitvi, pozoren tudi na »zanko za« (zanka for), ko se premikamo v desno oz. levo. Saj v zanki za premik v desno, povečujemo števec ``i`` za »+1« od 1 in vse do 15; pri zanki, ko se premikamo v levo, pa se števec ``i`` zmanjšujemo od 15 proti 1 za »-1«.

## KOŠARKAŠKI TRENING

SŠ NAPREDNI

Na posameznem igrišču so trije koši in tri žoge. Košarkarjeva naloga je, da žogo zabije v tisti koš, ki je najbližje izbrani žogi. Torej je potrebno košarkarja usmeriti tako, da bo vsako žogo zabil v svoj koš. Vendar pazi! Koši in žoge so na obeh igriščih, na različnih mestih.

Namig: Najprej poišči koš. Ko si na košu pokliči funkcijo "zabij\_kos", ki pregleda okolico koša.



[Povezava do naloge](#)

## Rešitev

Košarkarja premikamo po celotnem poligonu in sproti preverjamo, če smo na košu.



Košarkarja premikamo po iskalnem poligonu, ki je velikosti 5x5 in preverjamo, če smo našli iskano žogo. Če smo košarkarsko žogo našli, jo seveda pobereмо.



Košarkarja premaknemo v levi zgornji kot »iskalne površine«.

Iz desnega spodnjega kota »iskalne površine« košarkarja premaknemo nazaj do koša ter vanj zabijemo žogo, ki smo jo medtem pobrali.



## Ideja reševanja

Reševanja naloge Košarkarski trening se lahko lotimo na naslednji način:

1. Sprva je potrebno sprogramirati, da se košarkar premika po vrsticah, po celotnem poligonu, torej od zgornjega levega kota in vse do spodnjega (desnega) kota poligona. Medtem ko se košarkar premika po vrstici (v desno), pa naj tudi preverja, "če je morda na košu". Ko pride na konec vrstice, pa naj se premakne nazaj na izhodišče trenutne vrstice in nato še navzdol, v novo vrstico.
2. V kolikor pa košarkar med premikanjem po poligonu naleti na koš, pa kličemo funkcijo `zabij\_kos`.

V funkciji `zabij\_kos` pa je potrebno določiti in sprogramirati naslednje:

- a. Pred programiranjem je potrebno določiti iskalno površino, po kateri bo košarkar iskal žogo okoli koša; v našem primeru smo se odločili, da je iskalna površina iskanja žoge velikosti 5x5 (glej skico spodaj).
- b. Ko smo določili iskalno površino, pa košarkarja premaknemo v levi zgornji kot »le-te« in nato sistematično preiščemo celotno iskalno površino. V kolikor košarkar, med iskanjem, naleti na žogo pa jo takrat pobere.
- c. Ko pa pride košarkaš »v desni spodnji kot« iskalne površine, pa se vrne nazaj na koš in vanj zabije žogo, ki jo je medtem našel.

Slikovna ponazoritev funkcije `zabij\_kos` pa je tudi ponazorjena na spodnji skici.

