



PIŠEK 2019/20

2. poskusno (spletno) tekmovanje

Naloge in rešitve

V0.9

Januar 2021

Izbor, priredba in preoblikovanje tekmovalnih nalog: Programski svet tekmovalnega sistema

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

KAZALO NALOG

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

ZAJČEK SE PASE 1.....	1
SLASTNI KOVANČKI.....	2
PIŠEK GRE ČEZ REKO.....	3
ROBOT IŠČE VTIČNICO.....	4
HOJA PO SLEDI.....	5

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

PRENAŠANJE PAKETOV.....	6
ŠOPEK.....	7
PIŠEK, NARIŠI MI	8
PIŠEK ZBIRA 2-TISOČAKE	9
PO SLEDEH DO ZRN	10

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

SPREHOD GLEDE NA ŠTEVILA	12
TRIKOTNIKI.....	14
PIŠEK, NARIŠI MI	16
PIŠEK PRIŽIGA LUČKE	17
PIŠKOV NAJVIŠJI VRH.....	18
ROBOT V LABIRINTU.....	19

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

ŠOPEK.....	20
PIŠEK DODA PREBRANO ŠTEVILO.....	21
ROBOT ZLAGA OBLIKE	23
LUKA IGRA KOŠARKO.....	25
LARINA VETERNICA.....	27
PIŠEK RIŠE GOSENICO	29

Tekmovalna kategorija: SŠ ZAČETNIKI

HOJA PO SLEDI.....	31
--------------------	----

PIŠEK PRIŽIGA LUČKE	32
PIŠKOV NAJVIŠJI VRH	33
POLAGANJE PLOŠČIC	34
PALINDROM	35
Tekmovalna kategorija: SŠ NAPREDNI	
PIŠEK RIŠE VZORCE	37
OBILNA ŠTEVILA	38
PIŠEK DODA PREBRANO ŠTEVILO	40
PALINDROM	42
KRIŽANKA	44
Tekmovalna kategorija: SŠ POZNAVALCI	
POLAGANJE PLOŠČIC	46
PALINDROM	47
MONGOLSKI VZORCI	49
UREJANJE ŠTEVIL	51
KRIŽANKA	53
SPIRALA DO SIRA	55

ZAJČEK SE PASE 1

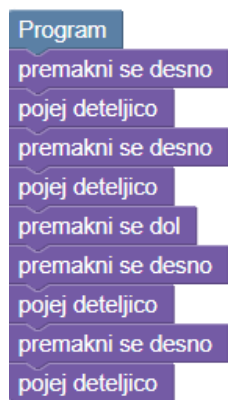
OŠ 4. – 6. r. ZAČETNIKI

Uredi zajčkova navodila tako, da bo sit. To pomeni, da mora pojesti vse deteljice!



[Povezava do naloge](#)

Rešitev



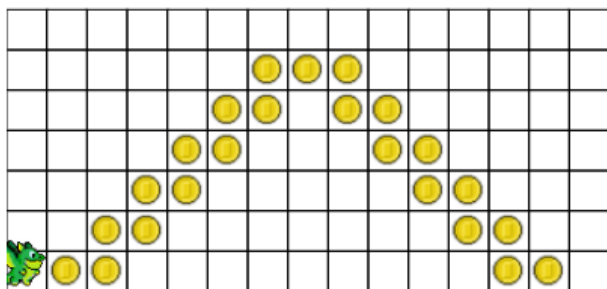
Ideja reševanja

Vsi ustrezni delčki so že na voljo na delovni površini. Urediti jih je potrebno le v pravilni vrstni red, torej določiti ustrezno zaporedje ukazov.

SLASTNI KOVANČKI

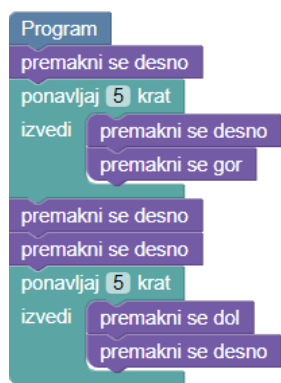
OŠ 4. – 6. r. ZAČETNIKI

Zmajček vidi kovance in si oblizne gobček! Vodi ga do vseh kovancev. Z delčkom »ponavljaljaj« lahko isti ukaz ponoviš večkrat.



Povezava do naloge

Rešitev



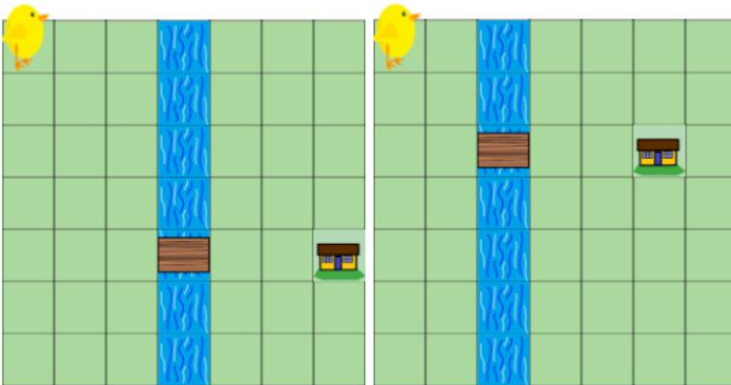
Ideja reševanja

To nalogo bi lahko rešili zgolj z osnovnim znanjem zaporedja ukazov, kar pa bi bilo zelo zamudno. Ko najdemo vzorec, ki se v kodi ponavlja, lahko uporabimo preprosto enojno zanko.

PIŠEK GRE ČEZ REKO

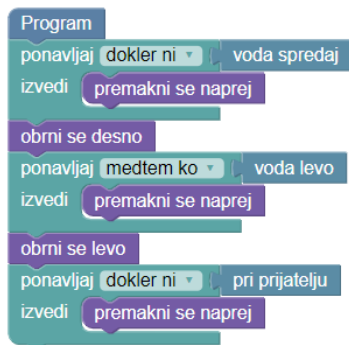
OŠ 4. – 6. r. ZAČETNIKI

Pišek se odpravi k prijatelju, ki stanuje na drugi strani reke. Najprej mora poiskati most, ga prečkati, nato pa je v treh korakih pri njem. Pozor: most je lahko na različnih mestih.



[Povezava do naloge](#)

Rešitev



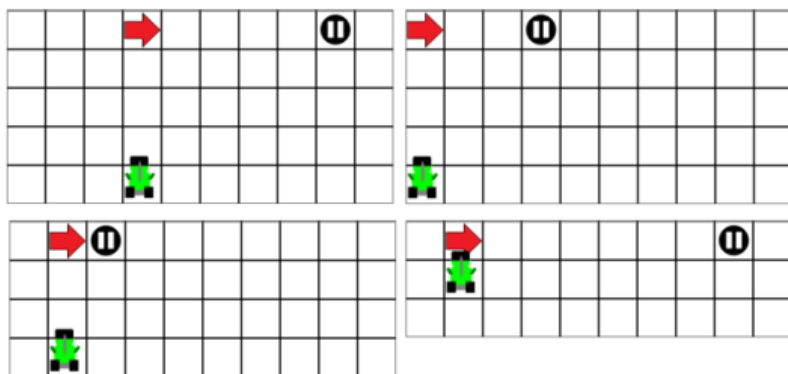
Ideja reševanja

Ker imamo v tej nalogi dva testna primera in se tako reka kot most lahko nahajata na različnih mestih, moramo uporabiti senzorje. Tretjo zanko v rešitvi pa lahko nadomestimo tudi z zaporedjem treh ukazov »premakni se naprej«, ker je cilj od mostu vedno oddaljen za tri premike.

ROBOT IŠČE VTIČNICO

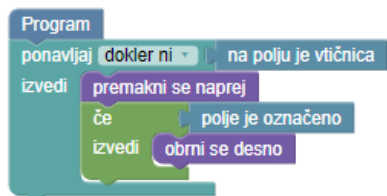
OŠ 4. – 6. r. ZAČETNIKI

Pomagaj Pišku in zloži delčke tako, da bo program deloval pravilno. A pozor! Program mora biti tak, da bo deloval za vse testne primere. Zato si oglej vse štiri in premisli, kaj je treba postoriti!



[Povezava do naloge](#)

Rešitev

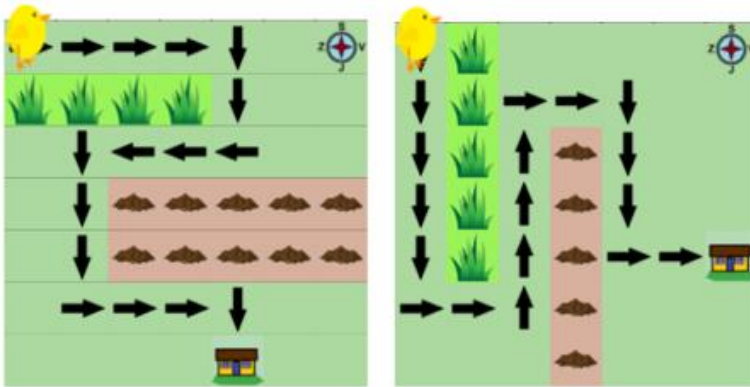


Ideja reševanja

Program mora univerzalno rešiti vse štiri testne primere, kar dosežemo z uporabo senzorjev. Ker rdeča puščica vedno kaže v desno, lahko uporabimo zgolj en pogojni stavek. Naloga dopušča uporabo zgolj šestih delčkov, zato je to edina možna rešitev.

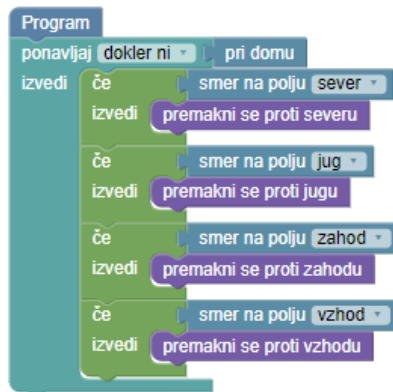
OŠ 4. – 6. r. ZAČETNIKI

Pišek bi rad prišel do svojega doma. Ker mora mimo zelenice in njive, si pomaga s smernimi puščicami. Sestavi program, ki bo Piška pripeljal domov. Pozor: isti program mora biti ustrezen za obe testni poti.



Povezava do naloge

Rešitev



Ideja reševanja

Program mora rešiti oba testna primera, zato moramo uporabiti vseh pet senzorjev. Premikanje v vseh štirih smereh neba določimo z uporabo pogojnih stavkov.

PRENAŠANJE PAKETOV

OŠ 4. – 6. r. NAPREDNI

Robot mora prenesti več paketov. Nosi lahko samo en paket naenkrat in paket lahko spusti samo, če ga je prej pobral. Na vsako označeno mesto prinesi samo en paket.



[Povezava do naloge](#)

Rešitev

```
Program
premakni se desno
premakni se desno
poberi paket
premakni se desno
pospravi paket
premakni se dol
premakni se levo
poberi paket
premakni se desno
premakni se desno
premakni se desno
pospravi paket
premakni se gor
premakni se levo
poberi paket
premakni se gor
premakni se levo
pospravi paket
```

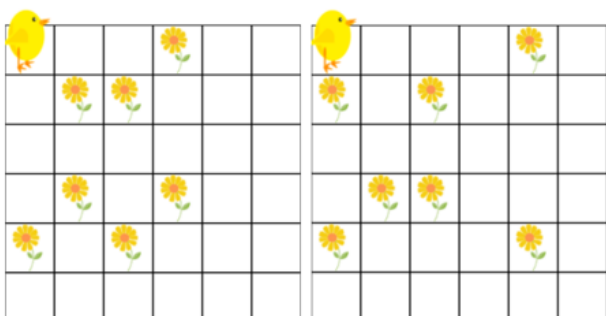
Ideja reševanja

Delčke moramo urediti v pravilno zaporedje. Možne so tudi druge rešitve.

ŠOPEK

OŠ 4. – 6. r. NAPREDNI

Pišek se veseli, da bo danes lahko obiskal babico, saj praznuje 70. rojstni dan. Na travniku za hišo ji bo nabral rože (za vsako desetletje eno). Napiši mu program, ki ga bo usmerjal. Ta mora biti tak, da deluje za oba testna primera.



Povezava do naloge

Rešitev



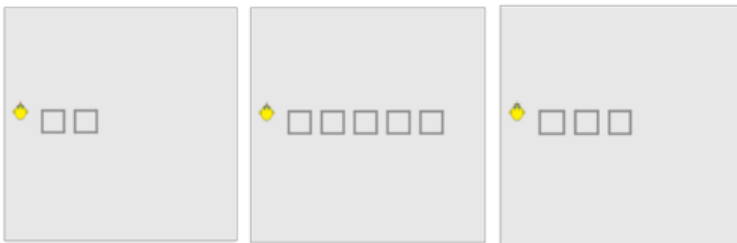
Ideja reševanja

Napisati moramo kodo, ki Piška vodi po mreži, med tem pa z uporabo pogojnega stavka in senzorja preverjamo lokacijo rož. Ker smo omejeni s številom delčkov, moramo uporabiti dvojno zanko. Pomembno je, da Pišek pregleduje zgolj prvih 5 vrstic, zadnje pa ne, ker bi sicer padel z mreže, v ta namen je zadnja vrstica prazna v obeh testnih primerih.

PIŠEK, NARIŠI MI ...

OŠ 4. – 6. r. NAPREDNI

Učiteljica poda Pišku naslednje navodilo: »Poglej program. V njem je že ukaz, ki prebere in shrani število kvadratov. Stopi 40 korakov desno in nariši toliko kvadratov, kot je prebrano število.« Stranica kvadrata naj bo velika 40 korakov, med kvadrati pa naj bo 20 korakov prostora.



Povezava do naloge

Rešitev



Ideja reševanja

Najprej moramo Piška postaviti na ustrezno začetno mesto, do katerega mora priti z dvignjenim pisalom. Kvadrat narišemo z uporabo preproste zanke. Z dvojno zanko pa narišemo več kvadratov, koliko pa nam pove vrednost, ki smo jo shranili v delčku »število«.

PIŠEK ZBIRA 2-TISOČAKE

OŠ 4. – 6. r. NAPREDNI

Pišek je navdušen planinec. Vsak teden gre v naravo in zabeleži višino vzpona. Ob koncu sezone želi v zeleno polje vpisati število 2-tisočakov. Napiši program.

	820	1478	2478	1802	2883	1918	2505	1004	2238	2132	1788		0
	887	820	2181	1534	2088	1154	2532	2558	1830	1834	1686		0

Povezava do naloge

Rešitev



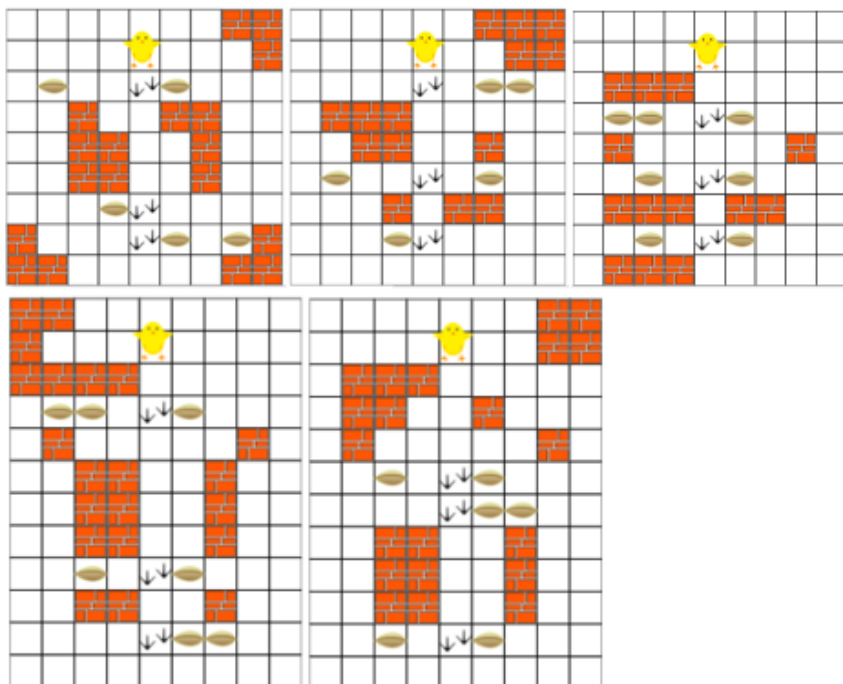
Ideja reševanja

Ustvariti moramo novo spremenljivko, ki jo lahko poljubno poimenujemo, mi smo jo »stevilo_2tisocakov«. To spremenljivko bomo povečali za 1 vsakič, ko bo številka polja večja ali enaka 2000, torej se je Pišek vzpel na 2-tisočaka. Na koncu ne smemo pozabiti izpisati vrednosti naše nove spremenljivke na zadnje zeleno polje.

PO SLEDEH DO ZRN

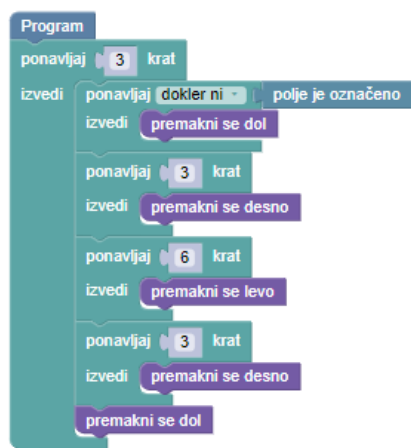
OŠ 4. – 6. r. NAPREDNI

Lačni Pišek se odpravi po poti stopinj. Napiši program, ki Piška vodi do zrn. Z delčkom »polje je označeno« v kategoriji senzor lahko preveriš, ali je Pišek našel sledi. Program mora biti tak, da deluje za vse testne primere.



[Povezava do naloge](#)

Rešitev



Ideja reševanja

Najprej si moramo dobro ogledati vseh pet testnih primerov. Hitro lahko ugotovimo, da imajo mreže v testnih primerih različno število vrstic, kar nakazuje, da bomo morali uporabiti zanko »ponavljaj dokler ni polje označeno«, kar nam svetuje že naloga sama. Ko se Pišek na sledi ustavi, vidimo, da ima zrna oddaljena največ tri korake v vsako smer, zato ga z uporabo preprostih zank lahko premaknemo v obe smeri in postavimo nazaj na polje s stopinjo. Na koncu pa moramo dodati še zunanjo zanko »ponavljaj 3-krat«, ker imajo vsi testni primeri natančno tri sledi.

SPREHOD GLEDE NA ŠTEVILA

OŠ 7. – 9. r. ZAČETNIKI

Pišek gre na sprehod. Njegova pot je odvisna od števil, ki jih sreča na poti. Če je prvo število večje, gre na rumeno označeno polje. Kadar je prvo število manjše od drugega, gre na zeleno polje. Takrat, ko pa sta števili enaki, konča svojo pot na modrem polju.



10	15				



23	8				



16	16				

[Povezava do naloge](#)

Rešitev



Ideja reševanja

Ustvariti moramo dve novi spremenljivki, ki ju lahko poljubno poimenujemo, mi smo ju »prvo_stevilo« in »drugo_stevilo«. Nato se postavimo na začetek spodnje vrstice. S pogojnimi stavki preverjamo relacije med obema številoma in se nato premaknemo do ustreznega polja. Če je prvo število manjše od drugega, tega ni potrebno preverjati, ker bomo ostali na istem mestu. Možne so tudi druge rešitve.

TRIKOTNIKI

OŠ 7. – 9. r. ZAČETNIKI

Sestavi program, ki prebere vse notranje kote trikotnika na vhodu in izpiše na izhod za vsakega ali je kot oster ali top.

Na vhodu (Input) se nahajajo tri števila v treh vrsticah. V vsaki vrstici je napisan en notranji kot trikotnika v stopinjah.

Program naj po vrsti kot so podani na izhod (Output) izpiše, kakšni so notranji koti. Če je kot oster, potem napiši "Kot je oster." Če je kot top, potem napiši "Kot je top." V tej nalogi ni noben notranji kot trikotnika enak 90 stopinj. Kot je oster, če je manjši od 90 stopinj, in top, če je večji od 90 stopinj.

Primer pravilnega delovanja programa:

Vhod:	Izhod:
57	Kot je oster.
103	Kot je top.
20	Kot je oster.

Input: 76 34 78	Output:	Input: 58 54 76	Output:
Input: 100 43 37	Output:		

[Povezava do naloge](#)

Rešitev



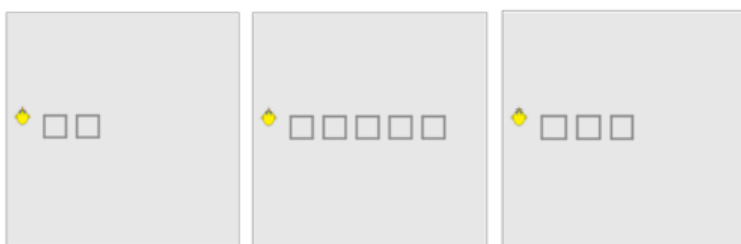
Ideja reševanja

Naloga ima že pripravljene tri spremenljivke: »kot1«, »kot2« in »kot3«, v katere shranimo vrednosti na vhodu. S pogojnimi stavki preverjamo ali so koti manjši oz. večji od 90 stopinj in izpišemo ustrezno besedilo.

PIŠEK, NARIŠI MI ...

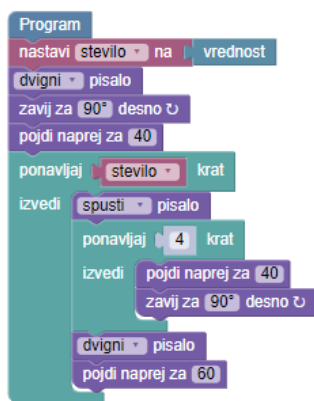
OŠ 7. – 9. r. ZAČETNIKI

Učiteljica poda Pišku naslednje navodilo: »Poglej program. V njem je že ukaz, ki prebere in shrani število kvadratov. Stopi 40 korakov desno in nariši toliko kvadratov, kot je prebrano število.« Stranica kvadrata naj bo velika 40 korakov, med kvadrati pa naj bo 20 korakov prostora.



Povezava do naloge

Rešitev



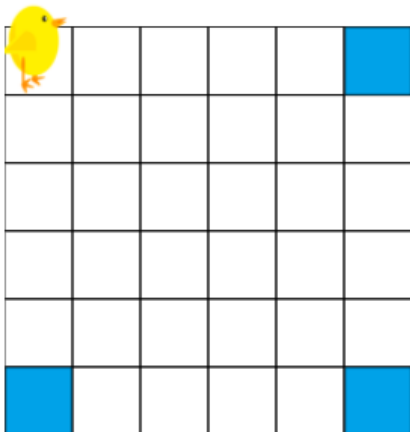
Ideja reševanja

Najprej moramo Piška postaviti na ustrezno začetno mesto, do katerega mora priti z dvignjenim pisalom. Kvadrat narišemo z uporabo preproste zanke. Z dvojno zanko pa narišemo več kvadratov, koliko pa nam pove vrednost, ki smo jo shranili v delčku »število«.

PIŠEK PRIŽIGA LUČKE

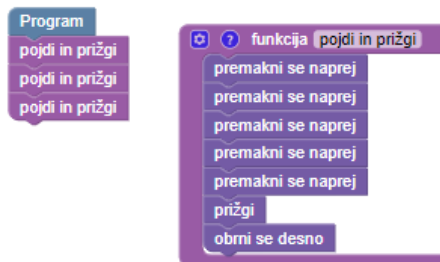
OŠ 7. – 9. r. ZAČETNIKI

Pišku so še posebej všeč naloge, kjer lahko narediš svoje ukaze - funkcije. Pomagaj Pišku prižgati vsa modra polja (jih spremeniti v rumena).



Povezava do naloge

Rešitev



Ideja reševanja

Na delovni površini imamo že pripravljeno ogrodje funkcije. V funkciji postavimo delčke v pravilno zaporedje. Funkcijo v programu pokličemo trikrat, ker mora Pišek prižgati tri lučke. Zaradi omejitve delčkov je to edina možna rešitev.

PIŠKOV NAJVIŠJI VRH

OŠ 7. – 9. r. ZAČETNIKI

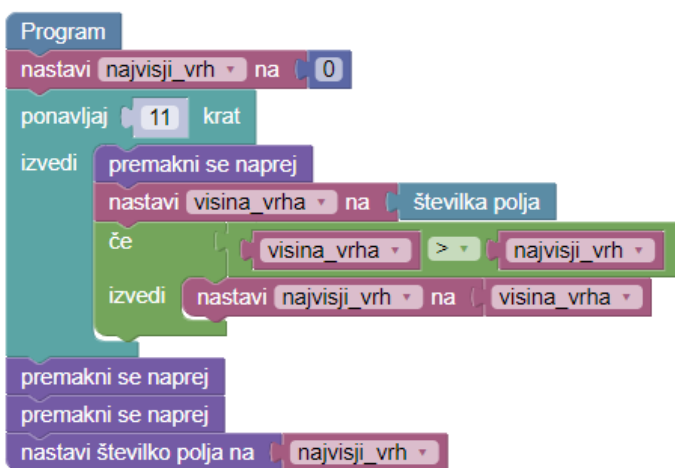
Piška vedno sprašujejo, kako visok je najvišji vrh, ki ga je preteklo leto obiskal. Pišek vsako leto obišče 11 vrhov. Zapisal si je njihove višine. S programom poišči največje število in ga vpiši v zeleno polje.

	820	1478	2476	1602	2863	1918	2505	1004	2236	2132	1768		0
---	-----	------	------	------	------	------	------	------	------	------	------	--	---

	667	820	2181	1534	2088	1154	2532	2558	1630	1634	1666		0
---	-----	-----	------	------	------	------	------	------	------	------	------	--	---

Povezava do naloge

Rešitev



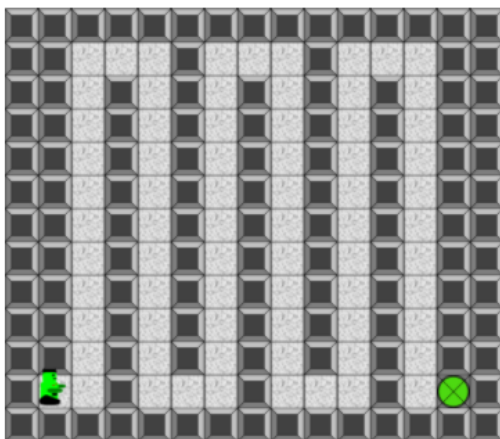
Ideja reševanja

Ustvariti moramo dve novi spremenljivki, ki ju lahko poljubno poimenujemo, mi smo ju »visina_vrha« in »najvisji_vrh«. Spremenljivko najvisji_vrh na začetku nastavimo na vrednost 0. S pomočjo zanke se 11-krat premaknemo naprej. Vsakič shranimo številko polja v spremenljivko visina_vrha, nato pa s pogojnim stavkom preverimo, ali je ta morda večja od do sedaj najvišjega vrha. Če je večja, to vrednost shranimo kot najvišji vrh. Na koncu ne pozabimo izpisati vrednosti najvišjega vrha v zadnje polje.

ROBOT V LABIRINTU

OŠ 7. – 9. r. ZAČETNIKI

Robot bi rad prišel do zelenega polja. Uporabi zanko v zanki.



[Povezava do naloge](#)

Rešitev



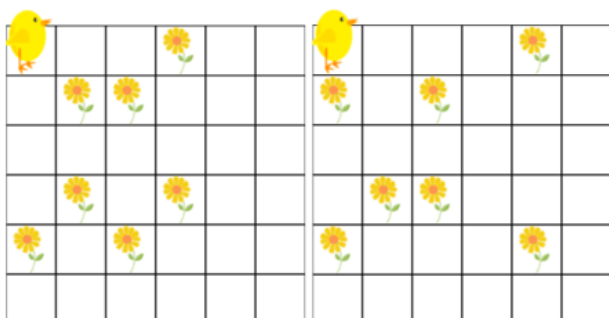
Ideja reševanja

Naloga ima omejitve delčkov, zato si moramo zelo dobro ogledati testni primer in poiskati vzorec, ki se ponavlja. Uporabimo dvojno zanko.

ŠOPEK

OŠ 7. – 9. r. NAPREDNI

Pišek se veseli, da bo danes lahko obiskal babico, saj praznuje 70. rojstni dan. Na travniku za hišo ji bo nabral rože (za vsako desetletje eno). Napiši mu program, ki ga bo usmerjal. Ta mora biti tak, da deluje za oba testna primera.



Povezava do naloge

Rešitev



Ideja reševanja

Napisati moramo kodo, ki Piška vodi po mreži, med tem pa z uporabo pogojnega stavka in senzorja preverjamo lokacijo rož. Ker smo omejeni s številom delčkov, moramo uporabiti dvojno zanko. Pomembno je, da Pišek pregleduje zgolj prvih 5 vrstic, zadnje pa ne, ker bi sicer padel z mreže, v ta namen je zadnja vrstica prazna v obeh testnih primerih.

PIŠEK DODA PREBRANO ŠTEVILO

OŠ 7. – 9. r. NAPREDNI

Pišek rad ureja števila od najmanjšega do največjega. Dve števili sta že urejeni, za njima pa je število 0. Pišek prebere število, ki je pred njim in ga vrine na ustrezno mesto (vrivanje pomeni, da mora mora števila premakniti).

Ko opravi nalogo, morajo biti števila na četrtem, petem in šestem kvadratu urejena, števila 0 pa ni več ...



25		12	34	0	
----	--	----	----	---	--



16		23	38	0	
----	--	----	----	---	--



25		5	12	0	
----	--	---	----	---	--

[Povezava do naloge](#)

Rešitev



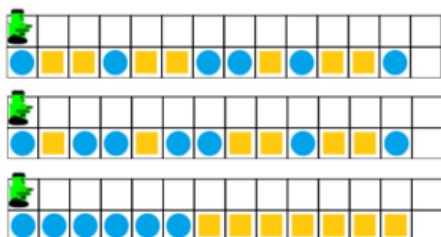
Ideja reševanja

Ustvariti moramo tri nove spremenljivke, ki jih lahko poljubno poimenujemo, mi smo jih »stevilo_novo«, »stevilo1« in »stevilo2«. Premaknemo se do ustreznih polj in shranimo številke polja v naše nove spremenljivke. Nato s pogojnimi stavki preverjamo relacije med števili. Ker vemo, da je vrednost spremenljivke »stevilo1« manjša od vrednosti »stevilo2« nam njunih relacij ni potrebno preverjati. Najprej preverimo relacijo med novim in 2. številom ter številki v poljih ustrezno premaknemo. Na koncu pa preverimo še relacijo med novim in 1. številom in po potrebi spremenimo številke polj. Možne so tudi druge rešitve.

ROBOT ZLAGA OBLIKE

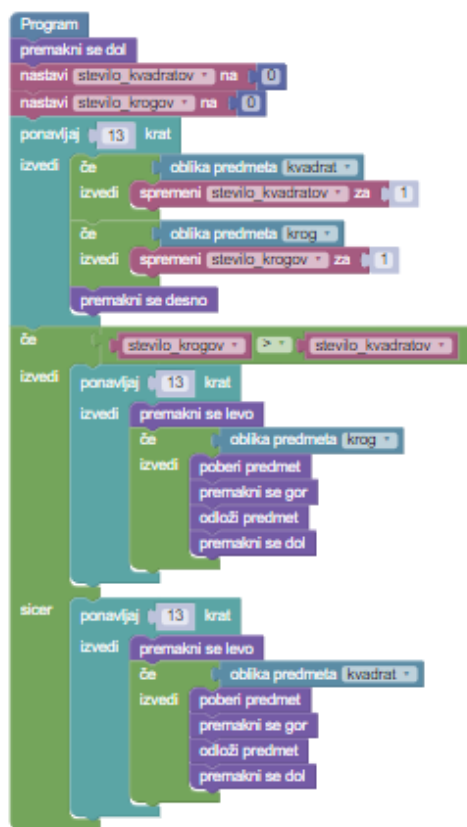
OŠ 7. – 9. r. NAPREDNI

Robota sprogramiraj tako, da bo najprej ugotovil, katera oblika je bolj pogosta v spodnji vrstici (je več krogov ali kvadratov). Potem naj objekte te oblike premakne eno vrstico višje. Tako mora v prvem testu robot premakniti kvadrate, ker jih je več kot krogov, pri drugem testu pa je več krogov in mora premakniti te!



[Povezava do naloge](#)

Rešitev



Ideja reševanja

Če sledimo navodilom, lahko ugotovimo, da moramo najprej preveriti, katerih likov je več, zato najprej ustvarimo dve novi spremenljivki, ki ju lahko poljubno poimenujemo, mi smo ju »stevilo_kvadratov«, »stevilo_krogov«. Z uporabo zanke se premikamo po mreži in na vsakem polju preverimo, kateri lik je na njem ter nato povečamo vrednost ustrezne spremenljivke za ena. Z uporabo pogojnega stavka primerjamo vrednosti obeh spremenljivk. Nazaj se premikamo z uporabo zanke med tem pa preverjamo tip lika na polju in ga po potrebi premaknemo v zgornjo vrstico. Namesto delčka »Če, izvedi – sicer« lahko uporabimo tudi dva preprosta pogojna stavka in v drugem preverjamo, ali je število kvadratov večje od števila krogov. Možnih je več rešitev.

LUKA IGRA KOŠARKO

OŠ 7. – 9. r. NAPREDNI

Luka trenira košarko. V tabelo si zapisuje, koliko točk je zbral na vsaki tekmi. Ko je v tabeli 0, pomeni, da Luka ni igral. Teh iger pri računanju povprečja ne bomo upoštevali. Sestavi program, ki prebere Lukine točke na zadnjih 10 tekmah in izpiše povprečno zbrano število točk.

0 20 0 30 33 0 24 37 31 21	Output:	31 0 25 20 40 0 29 21 38 36	Output:
29 0 0 34 37 37 29 35 39 40	Output:		

[Povezava do naloge](#)

Rešitev



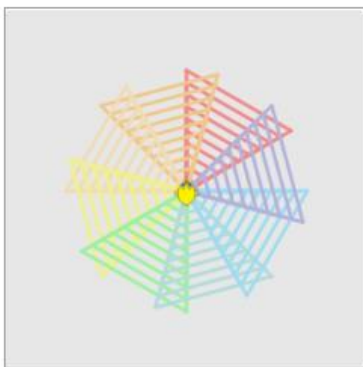
Ideja reševanja

Za rešitev te naloge potrebujemo tri spremenljivke, ki pa so vse že vnaprej pripravljene med dovoljenimi delčki. Na začetku moramo spremenljivki »skupno« in »stevilo_iger« nastaviti na vrednost 0, ker ju bomo nato spreminjali. Z uporabo zanke se premikamo po seznamu na vhodu. Vsakič nastavimo vrednost spremenljivke »tocke« na vrednost na vhodu in jo nato prištejemo skupnim točkam. Če vrednost spremenljivke »tocke« ni enaka nič, potem to igro lahko upoštevamo za računanje povprečja in zato spremenimo vrednost spremenljivke »stevilo_iger« za ena. Na koncu izpišemo povprečje točk, ki ga izračunamo kot količnik spremenljivk »skupno« in »stevilo_iger«. Možnih je več rešitev.

LARINA VETERNICA

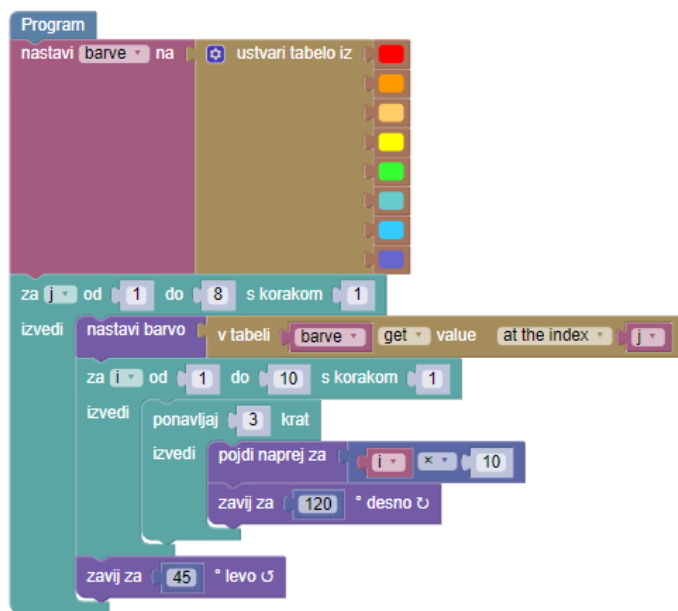
OŠ 7. – 9. r. NAPREDNI

Pišek bi rad Lari narisal vetrnico kakršna je narisana spodaj. Spremeni program tako, da bo Pišek narisal pravilno vetrnico.



[Povezava do naloge](#)

Rešitev



Ideja reševanja

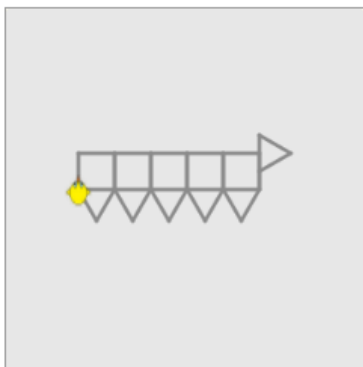
Program moramo zgolj popraviti. Najprej popravimo prvo zanko, in sicer potrebujemo vrednosti spremenljivke j od 1 do 8, ker imamo 8 različnih barv in ne zgolj 5. Nato popravimo zanko v ponavljalj 3-krat, ker Pišek riše trikotnike in ne štirikotnike. V zanki tudi popravimo zavoj v desno za 120 stopinj, ker je notranji kot enakostraničnega trikotnika 60 stopinj. Na koncu pa popravimo še vrednost zadnjega zavoja v levo za 45 stopinj, ker je polni kot 360 stopinj, imamo 8 barv, torej $360 \text{ stopinj} / 8 = 45 \text{ stopinj}$.

PIŠEK RIŠE GOSENICO

OŠ 7. – 9. r. NAPREDNI

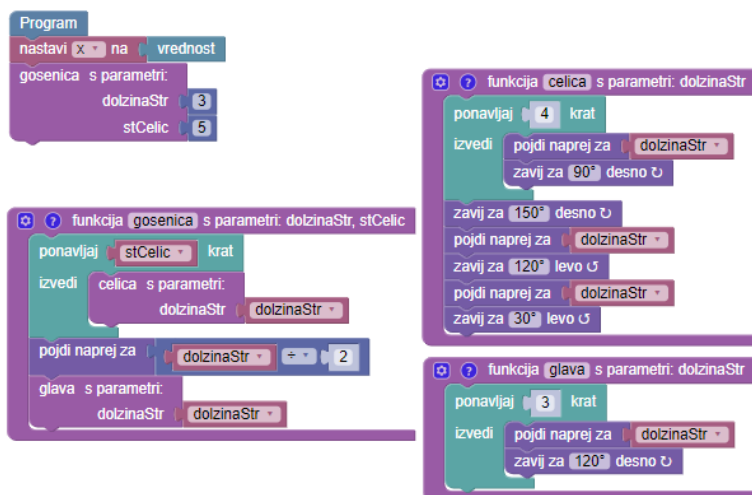
Piščanci radi iščejo gosenice. Če jih ne najdejo, jih narišejo. Vsaka gosenica ima več celic in trikotno glavo. Vsaka celica je sestavljena iz kvadrata in trikotnika. Kvadrat in trikotnik v celici ter trikotnik v glavi imata enako dolžino stranice.

Piščanec je hotel narisati gosenico na sliki, pa mu žal ni uspelo. V programu so napake. Poišči jih in popravi tako, da bo narisana taka gosenica, kot je označeno.



Povezava do naloge

Rešitev

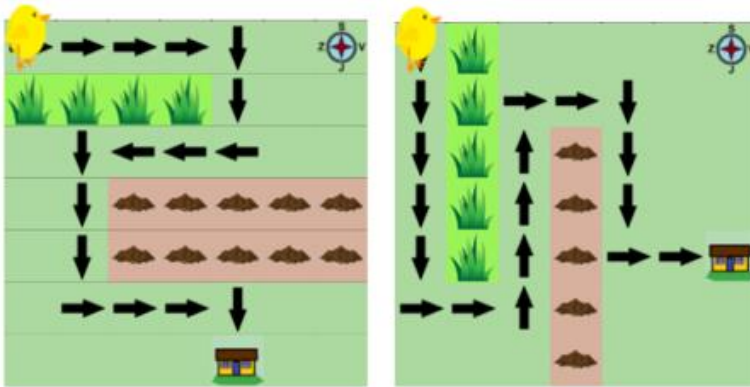


Ideja reševanja

Program moramo zgolj popraviti. Obe napaki se nahajata v funkciji celica. Najprej moramo popraviti zanko ponavljaj 4-krat, ker Pišek riše štirikotnike. Nato pa spremenimo še prvi zavoj v levo za 120 stopinj, ker noge gosenice sestavljajo enakostranični trikotniki.

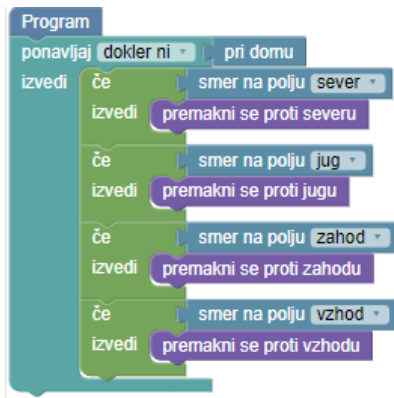
SŠ ZAČETNIKI

Pišek bi rad prišel do svojega doma. Ker mora mimo zelenice in njive, si pomaga s smernimi puščicami. Sestavi program, ki bo Piška pripeljal domov. Pozor: isti program mora biti ustrezen za obe testni poti.



Povezava do naloge

Rešitev



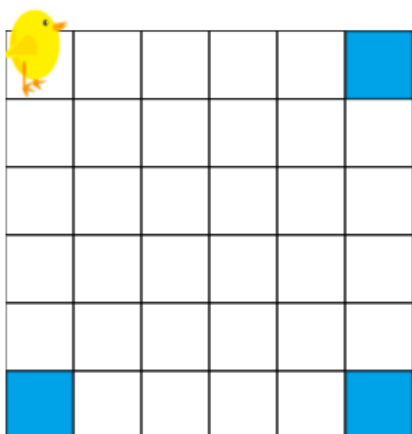
Ideja reševanja

Program mora rešiti oba testna primera, zato moramo uporabiti vseh pet senzorjev. Premikanje v vseh štirih smereh neba določimo z uporabo pogojnih stavkov.

PIŠEK PRIŽIGA LUČKE

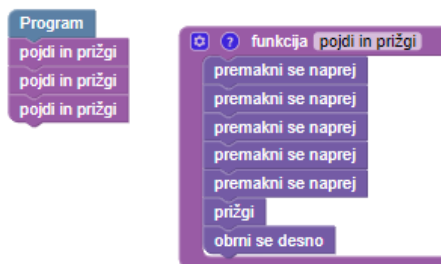
SŠ ZAČETNIKI

Pišku so še posebej všeč naloge, kjer lahko narediš svoje ukaze - funkcije. Pomagaj Pišku prižgati vsa modra polja (jih spremeniti v rumena).



[Povezava do naloge](#)

Rešitev



Ideja reševanja

Na delovni površini imamo že pripravljeno ogrodje funkcije. V funkciji postavimo delčke v pravilno zaporedje. Funkcijo v programu pokličemo trikrat, ker mora Pišek prižgati tri lučke. Zaradi omejitve delčkov je to edina možna rešitev.

PIŠKOV NAJVIŠJI VRH

SŠ ZAČETNIKI

Piška vedno sprašujejo, kako visok je najvišji vrh, ki ga je preteklo leto obiskal. Pišek vsako leto obišče 11 vrhov. Zapisal si je njihove višine. S programom poišči največje število in ga vpiši v zeleno polje.

	820	1478	2476	1602	2863	1918	2505	1004	2236	2132	1768		0
---	-----	------	------	------	------	------	------	------	------	------	------	--	---

	667	820	2181	1534	2088	1154	2532	2558	1630	1634	1666		0
---	-----	-----	------	------	------	------	------	------	------	------	------	--	---

Povezava do naloge

Rešitev



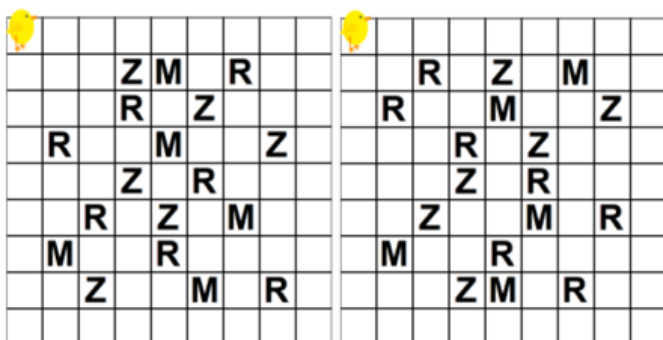
Ideja reševanja

Ustvariti moramo dve novi spremenljivki, ki ju lahko poljubno poimenujemo, mi smo ju »visina_vrha« in »najvisji_vrh«. Spremenljivko najvisji_vrh na začetku nastavimo na vrednost 0. S pomočjo zanke se 11-krat premaknemo naprej. Vsakič shranimo številko polja v spremenljivko visina_vrha, nato pa s pogojnim stavkom preverimo, ali je ta morda večja od do sedaj najvišjega vrha. Če je večja, to vrednost shranimo kot najvišji vrh. Na koncu ne pozabimo izpisati vrednosti najvišjega vrha v zadnje polje.

POLAGANJE PLOŠČIC

SŠ ZAČETNIKI

Pišek v kuhinji polaga keramične ploščice. Na tla je zapisal zahtevano barvo ploščice (R-rumena, M-modra, Z-zelena). Sestavi program, ki bo vodil Piška k ustrezni končni podobi tal. Polje pobarvaj z ukazom »pobarvaj z«. Zahtevano barvo prebereš s pomočjo senzorjev. Dobro si oglej oba testna primera.



[Povezava do naloge](#)

Rešitev



Ideja reševanja

Napisati moramo kodo, ki Piška vodi po mreži, med tem pa z uporabo pogojnega stavka in senzorja preverjamo, ali je polje označeno ter ga po potrebi pobarvamo. Ker smo omejeni s številom delčkov, moramo uporabiti dvojno zanko.

PALINDROM

SŠ ZAČETNIKI

Sestavi program, ki prebere besedo in ugotovi, ali je le ta palindrom. Palindrom je beseda (zaporedje znakov), ki se bere z obeh strani enako. Če je zapisana beseda palindrom, naj program izpiše DA, če ni, naj izpiše NE.

Vhod: Izhod:

ana DA

klovn NE

Input: cepec	Output:	Input: dovod	Output:
Input: anamarija	Output:		

[Povezava do naloge](#)

Rešitev



Ideja reševanja

Najprej si dobro ogledamo testne primere. Hitro lahko ugotovimo, da imata oba palindroma 5 črk, kar zelo olajša naše delo. Ustvarimo novo spremenljivko, ki jo lahko poljubno poimenujemo, mi smo jo poimenovali »besedilo«. Vanjo shranimo zapis iz vhoda. S pogojnimi stavki preverimo, ali sta prva in zadnja črka besedila enaki. Nato ponovimo postopek za drugo in predzadnjo črko. Če bi imeli več testnih primerov, bi morali poiskati univerzalno rešitev, ki je razložena pri rešitvi te naloge v naslednji kategoriji. Možne so tudi druge rešitve.

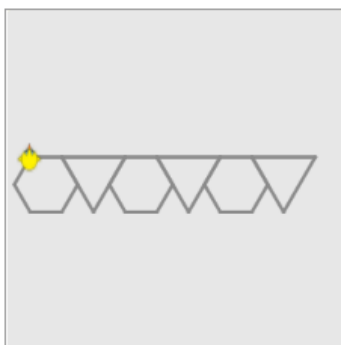
Pripombe dodal [BKT1]: Prosim za komentar. Je ta rešitev dopustna ali ne? Reši nalogo glede na naše teste, ni pa univerzalna. Tista rešitev je razložena v preostalih dveh kategorijah SŠ.

PIŠEK RIŠE VZORCE

SŠ NAPREDNI

Pišek se je zmedel pri risanju vzorcev. Pomagaj mu dokončati delo, da bo slika na koncu taka:

Stranice trikotnika so dolge 80 enot, stranice šestkotnika pa 40 enot.



[Povezava do naloge](#)

Rešitev



Ideja reševanja

Program moramo zgolj popraviti. Najprej popravimo zavoj v 2. zanki, in sicer zavije Pišek desno za 120 stopinj, ker riše enakostranični trikotnik. Nato pa moramo dodati en premik po vsaki zanki, da pride Pišek do izhodišča za začetek risanja naslednjega lika. Na koncu dodamo še eno zunanjo zanko, ker se vzorec ponovi trikrat.

OBILNA ŠTEVILA

SŠ NAPREDNI

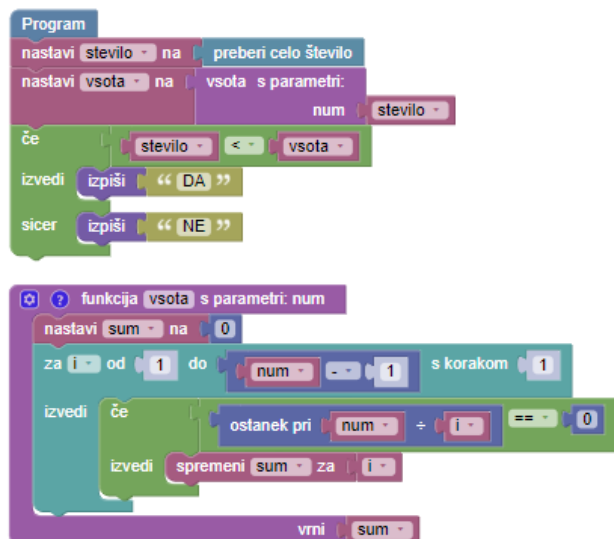
Janez je poizkušal napisati program, ki bo preveril, ali je vneseno število obilno. Pri tem je naredil nekaj napak. Popravi njegov program.

Obilno število je celo število, za katerega je vsota pravih deliteljev večja od števila samega. Število 12 je prvi primer obilnega števila. Njegovi pravi delitelji so 1, 2, 3, 4 in 6, kar je skupaj 16.

Input: 12	Output:	Input: 14	Output:
Input: 20	Output:	Input: 34	Output:
Input: 36	Output:	Input: 77	Output:
Input: 66	Output:	Input: 1	Output:

[Povezava do naloge](#)

Rešitev



Ideja reševanja

Program moramo zgolj popraviti. Najprej popravimo pogojni stavek v programu, saj mora biti vrednost spremenljivke »stevilo« manjša od vrednosti »vsota«. Nato pa moramo v funkciji vsota dodati še pogojni stavek, s katerim preverjamo, ali je število shranjeno v spremenljivki »i« pravi delitelj števila »num«, ki je parameter funkcije.

PIŠEK DODA PREBRANO ŠTEVILO

SŠ NAPREDNI

Pišek rad ureja števila od najmanjšega do največjega. Dve števili sta že urejeni, za njima pa je število 0. Pišek prebere število, ki je pred njim in ga vrine na ustrezno mesto (vrivanje pomeni, da mora mora števila premakniti).

Ko opravi nalogo, morajo biti števila na četrtem, petem in šestem kvadratu urejena, števila 0 pa ni več ...



25		12	34	0	
----	--	----	----	---	--



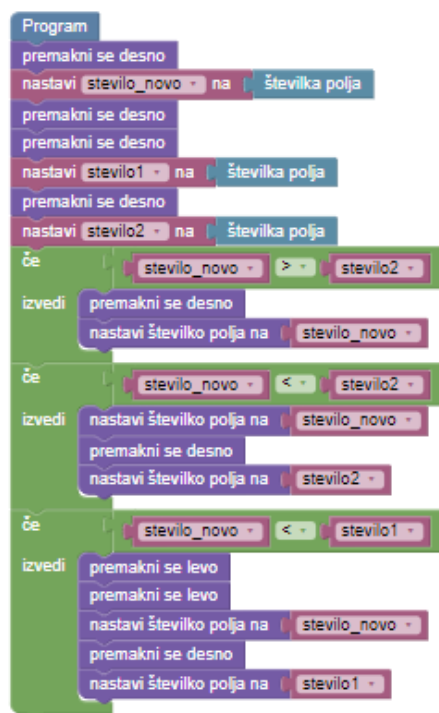
16		23	38	0	
----	--	----	----	---	--



25		5	12	0	
----	--	---	----	---	--

[Povezava do naloge](#)

Rešitev



Ideja reševanja

Ustvariti moramo tri nove spremenljivke, ki jih lahko poljubno poimenujemo, mi smo jih »stevilo_novo«, »stevilo1« in »stevilo2«. Premaknemo se do ustreznih polj in shranimo številke polja v naše nove spremenljivke. Nato s pogojnimi stavki preverjamo relacije med števili. Ker vemo, da je vrednost spremenljivke »stevilo1« manjša od vrednosti »stevilo2« nam njunih relacij ni potrebno preverjati. Najprej preverimo relacijo med novim in 2. številom ter številki v poljih ustrezno premaknemo. Na koncu pa preverimo še relacijo med novim in 1. številom in po potrebi spremenimo številke polj. Možne so tudi druge rešitve.

PALINDROM

SŠ NAPREDNI

Sestavi program, ki prebere besedo in ugotovi, ali je le ta palindrom. Palindrom je beseda (zaporedje znakov), ki se bere z obeh strani enako. Če je zapisana beseda palindrom, naj program izpiše DA, če ni, naj izpiše NE.

Vhod: Izhod:

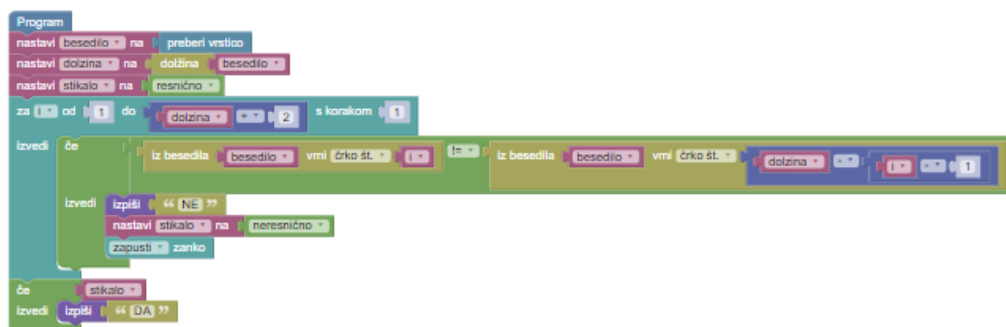
ana DA

klovn NE

Input: cepec	Output:	Input: dovod	Output:
Input: anamarija	Output:		

[Povezava do naloge](#)

Rešitev



Ideja reševanja

Ustvariti moramo tri nove spremenljivke, ki jih lahko poljubno poimenujemo, mi smo jih »besedilo«, »dolžina« in »stikalo«. V pogojnem stavku bomo postopoma preverjali različnost parov črk – prvo in zadnjo; drugo in predzadnjo itd. Zanka se bo izvajala tolikokrat, da bomo preverili vse pare zrcalno ležečih črk, torej polovico-dolžine-besede-krat. Ko bomo v zanki našli par črk, ki ni enak, bomo izpisali »NE«, nastavili stikalo za neresnično in prekinili izvajanje zanke. V primeru, ko se je zanka izvedla do konca brez prekinitve bo stikalo resnično in bomo izpisali »DA«.

Pripombe dodal [BKT2]: Prosim za mnenje o tej rešitvi. Je univerzalna, ampak ne vem, koliko so dijaki seznanjeni z uporabo stikala. Predlagate drugačno rešitev? Bi dopolnili razlago?

KRIŽANKA

SŠ NAPREDNI

Pišek preverja rešitev križank. Poskusil je napisati funkcijo, ki prebere besedo v križanki in jo zapiše na ustrezno mesto v tabeli besed. Besede bi morale biti v tabeli urejene glede na številko, ki je zapisana pred oziroma nad besedo. Lihe številke označujejo besede v stolpcu, sode pa v vrstici. Dopolni program tako, da bo izpisal tabelo besed v pravilnem vrstnem redu. Dobro poglej obe funkciji in razmisli! Pozor, tabela se začne z indeksom 0, v križanki pa je prva beseda označena z 1.

		3					
		S					
		O	7				
		K	8	N	O	V	
	1	O		O	5		
4	P	L	U	T	O	N	
		O		A			
		P		R		J	
2	L	E	P		K		
		R	6	A	T	A	

Input	Output

--	--

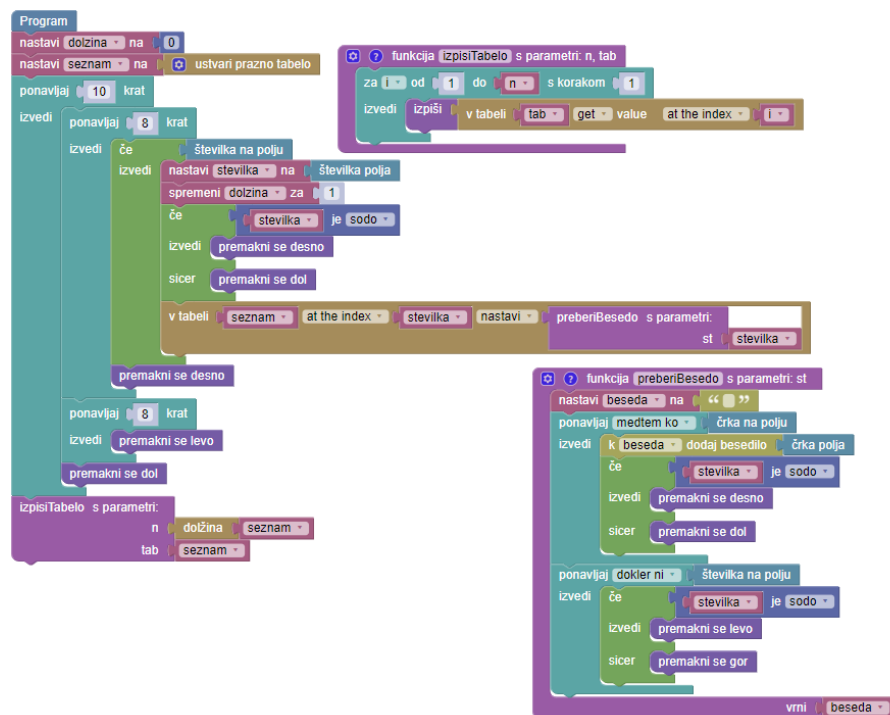
		1					
		B					
	3	O	5				
2	B	A	L	O	N		
	O	E		O			
4	B	E	Z	E	G		
	E	E		O			
	N	N	6	M	E	D	
				E			
				T			

Input	Output

--	--

[Povezava do naloge](#)

Rešitev



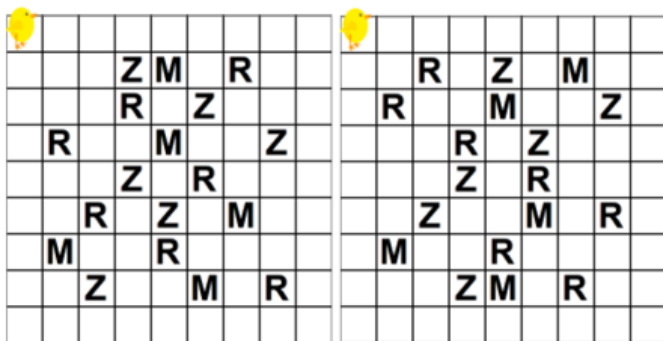
Ideja reševanja

Iz vnaprej podane kode prepoznamo, da imamo dvojno zanko, ki je namenjena iskanju po mreži dimenzije 10x8. Prepoznamo tudi pogojni stavek, ki preverja, ali se Pišek nahaja na številki. V tem primeru se kliče funkcija »preberi besedo«, ki še ni napisana. Ker nam navodila povedo, kako so besede zapisane glede na liho ali sodo število, moramo to število vnesti v funkcijo, ki bo nato določila premikanje v smeri levo/desno oz. gor/dol. Ker dolžine posamezne besede ne poznamo, moramo uporabiti zanko tipa »medtem ko«. Med branjem besede korakamo dokler je na polju črka. V obratno smer korakamo dokler ne pridemo do začetne številke. Na koncu programa moramo dodati še klic funkcije »klic tabele«, ki je nedokončana, saj očitno nima zanke.

POLAGANJE PLOŠČIC

SŠ POZNAVALCI

Pišek v kuhinji polaga keramične ploščice. Na tla je zapisal zahtevano barvo ploščice (R-rumena, M-modra, Z-zelena). Sestavi program, ki bo vodil Piška k ustrezni končni podobi tal. Polje pobarvaj z ukazom »pobarvaj z«. Zahtevano barvo prebereš s pomočjo senzorjev. Dobro si oglej oba testna primera.



[Povezava do naloge](#)

Rešitev



Ideja reševanja

Napisati moramo kodo, ki Piška vodi po mreži, med tem pa z uporabo pogojnega stavka in senzorja preverjamo, ali je polje označeno ter ga po potrebi pobarvamo. Ker smo omejeni s številom delčkov, moramo uporabiti dvojno zanko.

PALINDROM

SŠ POZNAVALCI

Sestavi program, ki prebere besedo in ugotovi, ali je le ta palindrom. Palindrom je beseda (zaporedje znakov), ki se bere z obeh strani enako. Če je zapisana beseda palindrom, naj program izpiše DA, če ni, naj izpiše NE.

Vhod: Izhod:

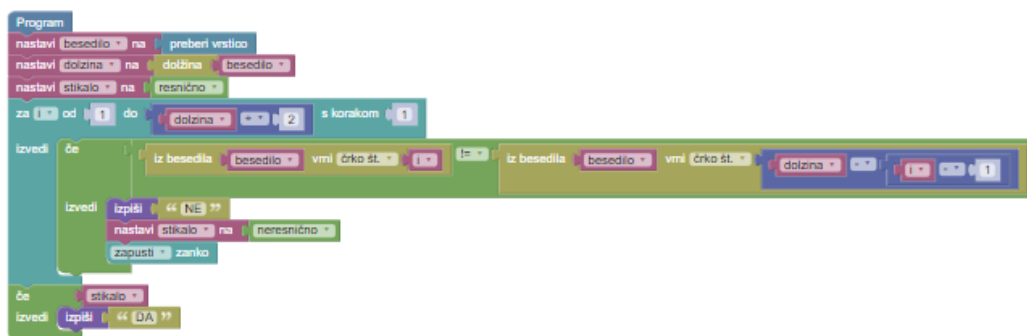
ana DA

klovn NE

Input: cepec	Output:	Input: dovod	Output:
Input: anamarija	Output:		

[Povezava do naloge](#)

Rešitev



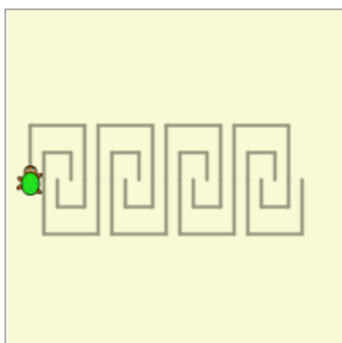
Ideja reševanja

Ustvariti moramo tri nove spremenljivke, ki jih lahko poljubno poimenujemo, mi smo jih »besedilo«, »dolžina« in »stikalo«. V pogojnem stavku bomo postopoma preverjali različnost parov črk – prvo in zadnjo; drugo in predzadnjo itd. Zanka se bo izvajala tolikokrat, da bomo preverili vse pare zrcalno ležečih črk, torej polovico-dolžine-besede-krat. Ko bomo v zanki našli par črk, ki ni enak, bomo izpisali »NE«, nastavili stikalo za neresnično in prekinili izvajanje zanke. V primeru, ko se je zanka izvedla do konca brez prekinitve bo stikalo resnično in bomo izpisali »DA«.

MONGOLSKI VZORCI

SŠ POZNAVALCI

Mongolski vzorci so posebno oblikovani ornamenti. Napiši program tako, da bo želva narisala enak mongolski vzorec, kot ga vidiš na sliki spodaj. Najkrajša stranica vzorca meri 30 enot. Pazi, število delčkov je omejeno.



[Povezava do naloge](#)

Rešitev



Ideja reševanja

Uporabimo preprosto zaporedje delčkov. Ker smo omejeni s številom delčkov, uporabimo zanke. Ker zlahka prepoznamo vzorec, ki se v sliki štirikrat ponovi, uporabimo dvojno zanko.

UREJANJE ŠTEVIL

SŠ POZNAVALCI

Katja zelo rada programira, zato je napisala funkcijo, ki sprejme tabelo števil, jih uredi po velikosti od najmanjšega do največjega in vrne urejeno tabelo. Nekega jutra je bila zelo zaspana in je datoteko s funkcijo odprla z napačnim urejevalnikom. Ta ji je nesramno razmetel delčke, ki sestavljajo funkcijo.

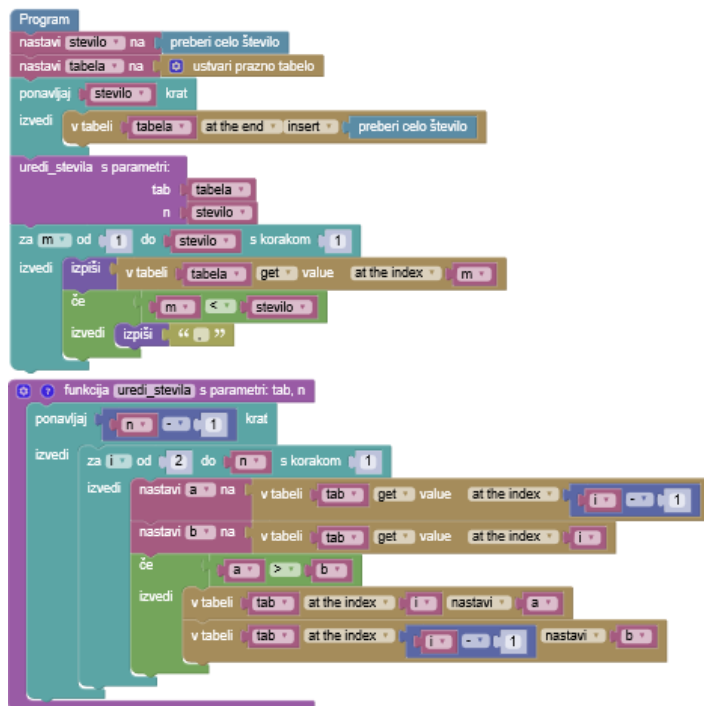
Na srečo so vsaj delčki testnega programa ostali v pravilnem vrstnem redu.

Preuredi dane vrstice funkcije »uredi_stevila(tab)« tako, da bodo vsi testni primeri ponovno delovali!

Input: 2 0 5	Output:	Input: 3 10 2 4	Output:
Input: 5 3 0 3 0 0	Output:	Input: 10 3 7 1 2 11 6 1 2 -3 0	Output:

[Povezava do naloge](#)

Rešitev



Ideja reševanja

Da bo program deloval, moramo pravilno urediti delčke v funkciji »uredi_stevila«. Nalogo lahko rešimo le tako, da zamenjujemo sosede v tabeli. Ena izmed zank je namenjena korakanju po vseh parih sosedov v tabeli. V posameznem koraku torej pravilno preuredimo vse izmed sosedskih parov. Sedaj pa moramo še ugotoviti kolikokrat moramo ta korak ponoviti. Temu je namenjena druga zanka, do katere lahko intuitivno pridemo, če se vprašamo; če je najmanjše število na zadnjem mestu, koliko sosedskih zamenjav potrebujemo da ga spravimo na prvo mesto. Zanki imata enako število ponovitev. Ugotovimo, da spremenljivka »i« kliče sosede in mora biti zato notranja zanka. Zamenjavo dveh sosedov napravimo tako, da njuni vrednosti shranimo v dveh spremenljivkah in jih, če nista urejeni, zamenjamo.

KRIŽANKA

SŠ POZNAVALCI

Pišek preverja rešitev križank. Poskusil je napisati funkcijo, ki prebere besedo v križanki in jo zapiše na ustrezno mesto v tabeli besed. Besede bi morale biti v tabeli urejene glede na številko, ki je zapisana pred oziroma nad besedo. Lihe številke označujejo besede v stolpcu, sode pa v vrstici. Dopolni program tako, da bo izpisal tabelo besed v pravilnem vrstnem redu. Dobro poglej obe funkciji in razmisli! Pozor, tabela se začne z indeksom 0, v križanki pa je prva beseda označena z 1.

		3					
		S					
		O	7				
		K	8	N	O	V	
	1	O		O	5		
4	P	L	U	T	O	N	
		O		A		O	
		P		R		J	
2	L	E	P		K		
		R	6	A	T	A	

Input	Output

--	--

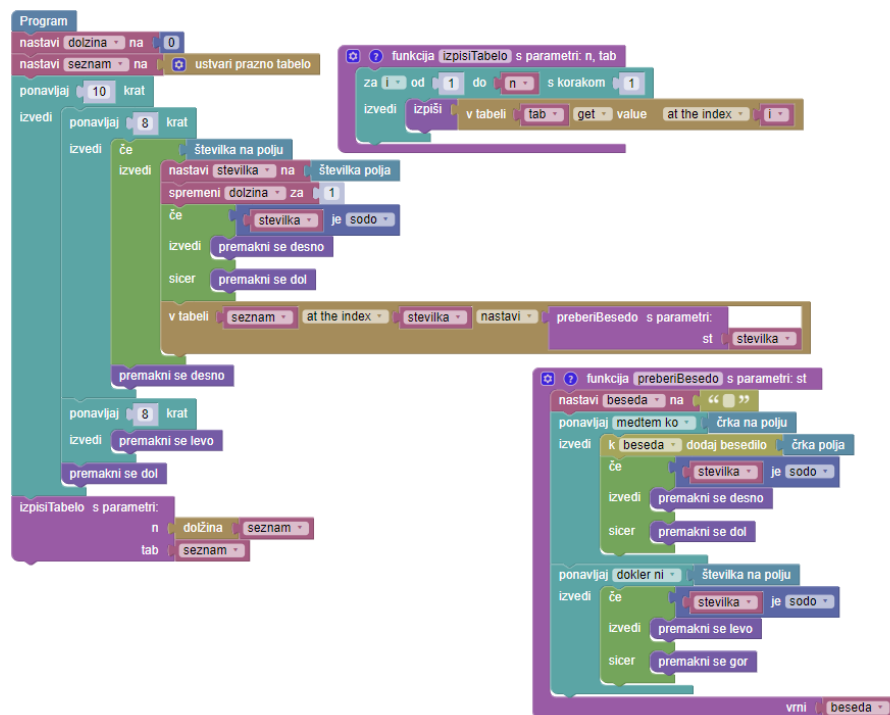
		1					
		B					
		O	5				
3		B	A	L	O	N	
2	B	A	L	O	N		
	O	E		O			
4	B	E	Z	E	G		
	E	E		O			
	N	N	6	M	E	D	
				E			
				T			

Input	Output

--	--

[Povezava do naloge](#)

Rešitev



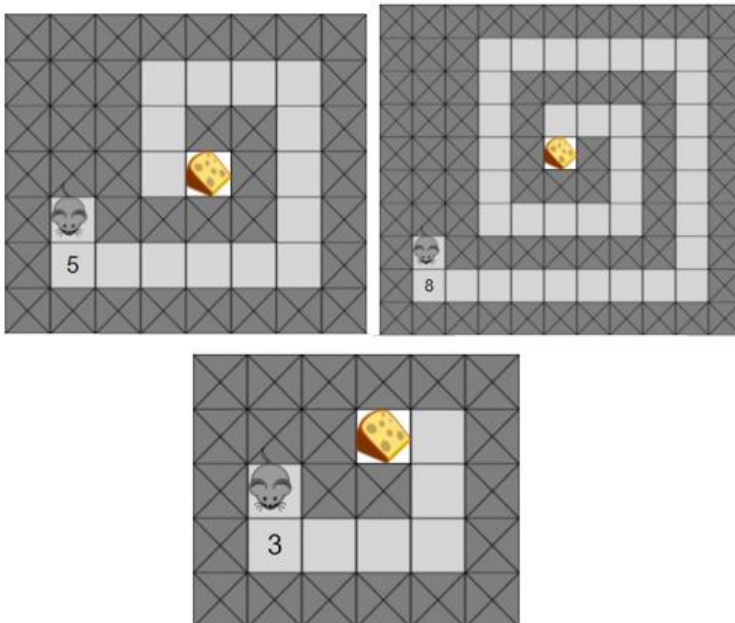
Ideja reševanja

Iz vnaprej podane kode prepoznamo, da imamo dvojno zanko, ki je namenjena iskanju po mreži dimenzije 10x8. Prepoznamo tudi pogojni stavek, ki preverja, ali se Pišek nahaja na številki. V tem primeru se kliče funkcija »preberi besedo«, ki še ni napisana. Ker nam navodila povedo, kako so besede zapisane glede na liho ali sodo število, moramo to število vnesti v funkcijo, ki bo nato določila premikanje v smeri levo/desno oz. gor/dol. Ker dolžine posamezne besede ne poznamo, moramo uporabiti zanko tipa »medtem ko«. Med branjem besede korakamo dokler je na polju črka. V obratno smer korakamo dokler ne pridemo do začetne številke. Na koncu programa moramo dodati še klic funkcije »klic tabele«, ki je nedokončana, saj očitno nima zanke.

SPIRALA DO SIRA

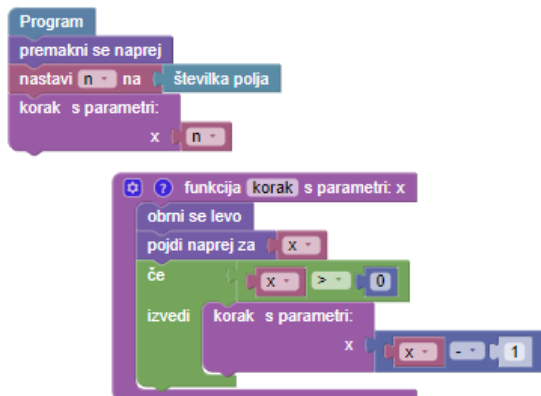
SŠ POZNAVALCI

Pomagaj miši, da pride do sira skozi različne labirinte v obliki spirale. Pri tem lahko uporabiš rekurzivno funkcijo.



[Povezava do naloge](#)

Rešitev



Ideja reševanja

Nalogo rešimo z rekurzivno funkcijo. Funkcija prejme (vsaj) en parameter, ta parameter se mora ob klicu funkcije spremeniti, funkcija mora poklicati samo sebe s spremenjeno vrednostjo parametra in na koncu moramo imeti še pogoj, kdaj funkcija neha klicati samo sebe (rekurzija se zaključi).