

Zadatak IGRA	Autor: Marija Gegić
---------------------	----------------------------

S obzirom na to da je svako polje u igri omeđeno crtama duljine 50 piksela, lako je uočiti da se svih 9 polja za igru jednostavno dobije crtanjem dviju okomitih crta duljine 150 piksela razmaknutih za 50 piksela, te crtanjem dviju vodoravnih crta duljine 150 piksela, razmaknutih ponovno za 50 piksela. Kako zadatak traži i crtanje križića i kružića, potrebno se pomaknuti u središte središnjeg polja za igru, odnosno pomaknuti se za 25 piksela od lijevog ruba tog središnjeg polja i za 25 piksela od donjeg ruba središnjeg polja. Zatim četiri puta ponovimo crtanje crte duljine 25 piksela, vraćanje u središte te okret od 90 stupnjeva, kako bismo nacrtali križić. Da nacrtamo kružić, potrebno se pomaknuti u središte polja koje se nalazi direktno desno od središnjeg. Njegovo se središte nalazi 50 piksela udaljeno od središta polja u kojem se nalazi križić, pa je potrebno kornjaču pomaknuti na odgovarajuću središnju poziciju i naredbom CIRCLE nacrtati kružnicu polumjera 20 piksela.

Potrebna znanja: osnovne naredbe za pomicanje kornjače, crtanje kružnice

Zadatak PIZZA	Autor: Frano Mihaljević
----------------------	--------------------------------

Za rješavanje zadatka bilo je potrebno nacrtati dvije kružnice (radijusa 100 i 115 piksela) te 8 puta ponoviti: crtanje ravne linije duljine 115, okret za šesnaestinu punog kuta, pomak unaprijed s podignutim perom za 60 piksela, crtanje kružnice radijusa 15 piksela, pomak unatrag i okret za šesnaestinu punog kuta.

Potrebna znanja: osnovne naredbe za pomicanje kornjače, crtanje kružnice

Zadatak SRCA	Autor: Marija Gegić
---------------------	----------------------------

Promotrimo najprije kako nacrtati jedno srce. Prije samog crtanja, potrebno se za 10 piksela udaljiti od središta ekrana, promijeniti boje crtanja i ispune na crvenu primjenom naredbi SETPC i SETFC te pravilno usmjeriti kornjaču kako bismo mogli nacrtati kvadrat stranice duljine 80 piksela. Nakon toga se potrebno bez ostavljanja traga pomaknuti u unutrašnjost kvadrata, kako bismo ga mogli ispuniti crvenom bojom. Nakon toga se pomaknemo na polovišta stranica koje su promjeri kružnica te nacrtamo kružnice čiji je polumjer 40 piksela, odnosno dvostruko manji od duljine stranica kvadrata. Kako bismo ispunili te kružnice, potrebno se pomaknuti u njihovu unutrašnjost, pazeći da se prije bojanja nalazimo na neobojanom pikselu. To možemo učiniti tako da se pomaknemo za neku malu udaljenost od središta kružnica te ih naredbom FILL ispunimo. Nakon toga se vratimo u središte ekrana, pravilno usmjerimo prije crtanja sljedećeg srca, te opisani postupak ponovimo ukupno četiri puta.

Potrebna znanja: osnovne naredbe za pomicanje kornjače, naredba FILL

Zadatak MAŠNA	Autor: Ivan Paljak
----------------------	---------------------------

Za 40% bodova na zadatku bilo je dovoljno nacrtati dva jednakokračna trokuta koja imaju zajednički vrh i osno su simetrični. Trokuti su u zadatku zadani duljinom osnovice i njezinom pripadajućom visinom pa ih možemo crtati sljedećim algoritmom:

```
pu rt 90 fd :a lt 90 pd fd :b/2 home
pu rt 90 fd :a rt 90 pd fd :b/2 home
pu lt 90 fd :a rt 90 pd fd :b/2 home
pu lt 90 fd :a lt 90 pd fd :b/2 home
```

Kada bismo prije crtanja ovih trokuta nacrtali kružnicu radijusa r koju smo ispunili crnom bojom, dobili bismo sliku mašne uz pogrešno obojenu kružnicu. Zatim ćemo kružnicu nacrtati nekom drugom bojom, njenu unutrašnjost ispuniti bijelom bojom i konačno nacrtati kružnicu crnom bojom.

Potrebna znanja: osnovne naredbe za pomicanje kornjače, naredba FILL

Zadatak UTRKA	Autor: Frano Mihaljević
----------------------	--------------------------------

Glavni problem zadatka bio je pronaći kut za koji se treba okrenuti da bismo nacrtali kružnicu radijusa a , odnosno kut na kojem će se nalaziti Filip i Goran u trenutku kad jedan od njih dvojice prođe drugog za puni krug.

Filip će za jednu minutu proći kut f , za pola minute kut $f/2$, a za n minuta kut $n \cdot f$, dok će Goran za n minuta proći kut $n \cdot g$. Ako je Filip brži od Gorana, Filip će proći gorana za puni krug kada je $n \cdot f - n \cdot g = 360$. Iz ovoga slijedi da je $n = 360 / (f - g)$. Već smo rekli da će nakon n minuta Filip proći kut $n \cdot f$. Ovaj izraz u trenutku kad Filip prođe Gorana za puni krug postaje $(360 / (f - g)) \cdot f$. Dakle, do tog trenutka Filip prođe kut $(360 / (f - g)) \cdot f$. Da bismo dobili kut u kojem se nalazi potrebno je još pribrojiti početni kut k pa je kut za koji se moramo okrenuti $(360 / (f - g)) \cdot f + k$. Primijetite da smo na isti način mogli dobiti kut koji prođe Goran, a budući da se Filip i Goran nalaze u istoj točki, taj postupak je jednako točan i rezultirat će jednakim konačnim usmjerenjem kornjače na ekranu. Ako je Goran brži od Filipa, izraz postaje malo drukčiji - zamijenjeni su umanjitelj i umanjitelj.

Potrebna znanja: osnovne naredbe za pomicanje kornjače, crtanje kružnice

Zadatak DOBRE**Autor: Marija Gegić**

Za osvajanje 10% bodova na ovome zadatku, dovoljno je bilo vratiti nepromijenjenu listu :l. Promotrimo sada jedno od mogućih rješenja koje bi osvojilo ukupno 70% bodova, a jednostavno se može modificirati za osvajanje svih 120 bodova. Kada bismo za svaku riječ znali odrediti je li dobra, dovoljno bi bilo proći po listi :l te za svaku riječ provjeriti je li dobra te, ako jest, spremi je u neku pomoćnu listu. Na kraju će te pomoćna lista biti traženo rješenje. Za određivanje je li neka riječ :r dobra, potrebno je za svaku podlistu iz liste :p proći po cijeloj riječi :r, te za svaka dva uzastopna znaka provjeriti jesu li jednaka elementima iz trenutne podliste za koju provjeravamo. Pritom ne smijemo zaboraviti uzeti u obzir oba poretka slova u podlistama. Ako postoji podlista liste :p čiji se elementi nalaze u uzastopnom poretku u riječi :r, onda riječ :r nije dobra. U suprotnom, riječ :r je dobra.

Za osvajanje preostalih 30% bodova, potrebno je listu koja sadrži sve dobre riječi sortirati po kriteriju opisanom u tekstu zadatka. To možemo učiniti primjenom nekog od poznatih algoritama sortiranja, ali možemo iskoristiti i funkciju SORT. Budući da bi funkcija SORT listu sortirala leksikografski, trebamo specificirati način na koji želimo da lista bude sortirana, tako da kao dodatan argument zadamo funkciju za usporedbu dvaju elemenata. Za implementacijske detalje pogledajte službeno rješenje.

Potrebna znanja: liste, sortiranje

Zadatak CONNECT**Autor: Antea Hadviger**

Prvi korak u rješavanju zadatka jest nacrtati prazno polje za igru. Uzevši u obzir promjere kružnica i razmake između njih, lako se izračuna da je visina pravokutnika koji omeđuje polje jednaka $n \cdot (2 \cdot r + d) + d$, a širina analogna. Pomoću dviju petlji crta se mreža kružnica, pazеći da razmak od ruba polja bude pravilan.

Za osvajanje 40% bodova na zadatku, nije bilo potrebno provjeravati u kojem trenutku igra završava, već samo simulirati odigrane poteze. U službenom rješenju, u listi :h, na početku popunjenoj nulama, zapisano je kolika je trenutna visina kojeg stupca, tj. koliko je pločica u njega dosad ubačeno. Na taj način ćemo znati koje polje ploče treba popuniti bojom u svakom koraku. Ako kornjača stoji u donjem lijevom kutu, pomaknemo je udesno tako da budemo na dnu stupca u kojeg se ubacuje pločica, te prema gore u ono polje po redu koje je prvo slobodno, što očitamo iz liste :h. Ovisno o parnosti poteza, polje u kojem se nalazimo obojimo plavom ili crvenom bojom te visinu stupca povećamo za 1. U službenom rješenju, povećanje elementa riješeno je pomoću funkcije SET :l :i :v koja u listi :l :i-ti član postavlja na vrijednost :v.

Za osvajanje svih bodova, nakon svakog je koraka potrebno provjeriti je li igra završena. Službeno rješenje koristi listu :c koja se sastoji od :n listi od :m elemenata, a u njoj je zapisano koje je polje obojeno kojom bojom, ako nije prazno. Na početku se cijela lista popuni oznakama koje označavaju prazno polje. Nakon svakog poteza, na mjestu gdje smo postavili pločicu ažuriramo boju u listi :c. Provjeravamo je li igra završena pomoću dviju dvostrukih petlji. Jedna provjerava ima li 4 uzastopne pločice iste boje u nekom stupcu, a druga radi isto za retke. Pomoću varijable :cnt brojimo koliko je trenutno uzastopnih pločica iste boje viđeno u nekom retku (stupcu). Ako pločica na kojoj smo trenutno nije prazna i iste je boje kao i prethodna, brojač povećamo za 1, a u suprotnom ga resetiramo, odnosno postavljamo na 1. Ako brojač u nekom trenutku dosegne vrijednost 4, igra je završena i prekidamo izvršavanje programa.

Potrebna znanja: osnovne naredbe za pomicanje kornjače, naredba FILL, petlje liste

Zadatak LABIRINT	Autor: Mihael Liskij
-------------------------	-----------------------------

S obzirom da je ovo bio interaktivni zadatak, za početak je bilo bitno shvatiti način na koji se robot kreće te kako njime upravljati danim funkcijama. Nakon toga je bilo potrebno osmisliti algoritam otkrivanja labirinta te ga u konačnici nacrtati, tj. podijeliti naš problem na logičko rješavanje labirinta i njegovo konačno crtanje nakon što smo ga u potpunosti otkrili.

Za kretanje robota bilo je potrebno koristiti priloženu pomoćnu funkciju. Kako naš robot ne vidi labirint, moramo koristiti neki drugi način za otkrivanje zidova. Dobra ideja jest prilikom micanja gledati dva slučaja: kada se uspije i kada se ne uspije pomaknuti. Prilikom uspješnog pomicanja, znamo da smo se pomaknuli u smjeru koji smo zatražili pa shodno tome možemo pomoću nekih globalnih varijabli pamti x i y poziciju robota u labirintu. Ako samo logički želimo saznati izgled labirinta, potpuno nam je svejedno ako za početne x i y koordinate uzmemo [-50 -23] ili [5 5], naš labirint će samo biti odmaknut od zamišljenog središta.

Nakon što smo razradili kako upravljati pozicijom robota, moramo smisliti način na koji ćemo labirint obići. Odabrani algoritam službenog rješenja je DFS (*depth first search*), rekurzivno pretraživanje u dubinu.

Pretpostavimo za početak da se nalazimo na polju [1 1] labirinta za koji ne znamo gdje se točno nalazimo. Također dopuštamo da nam pozicija polja u labirintu postane negativan broj. Dok provodimo algoritam, u neku pomoćnu listu pamtimo sva polja na kojima smo već bili. Tako na početku u toj pomoćnoj listi imamo samo poziciju [1 1]. U svakom trenutku se probamo pomaknuti u neki od četiri moguća smjera, što ima dva moguća ishoda:

- 1) Zapnemo - u tom slučaju ne moramo ništa raditi
- 2) Uspijemo se pomaknuti

Ako se uspijemo pomaknuti na neko susjedno polje, npr. [1 2], moramo provjeriti da nismo već bili na tom polju, a to možemo znati jer imamo pomoćnu listu sa svim pozicijama na kojima smo bili. Kako se pozicija [1 2] ne nalazi u toj pomoćnoj listi, mi je ubacujemo u nju pa nam je pomoćna lista sada [[1 1][1 2]]. Nakon što smo ubacili novu poziciju u tu listu, rekurzivno se širimo na to novo polje i ponavljamo algoritam. U slučaju da je to polje već bilo u pomoćnoj listi, ne širimo se na to polje.

Kako bismo osigurali da naš algoritam točno radi, potrebno je još nakon rekurzivnog poziva napraviti suprotni potez od onoga koji smo koristili da bi došli na to polje. Ovaj zadnji korak je općenito nužan uvjet da bi rekurzije točno radile: stanje programa na kraju rekurzije uvijek mora odgovarati stanju programa na početku rekurzije. Time si smanjujemo šansu za pogrešku prilikom implementacije. Iz ovoga slijedi da ako je naš robot krenuo iz [1 1] da će i završiti u [1 1].

Konačno iz pomoćne liste moramo rekonstruirati labirint. Za početak pronađemo najmanju x_{\min} koordinatu i najmanju y_{\min} koordinatu. Poznavajući njih možemo translirati cijeli labirint da počinje od [1 1] tako da za svako polje u pomoćnoj listi napravimo sljedeću transformaciju: $[x-x_{\min}+1 \ y-y_{\min}+1]$. Sada je samo preostalo nacrtati pravokutnu mrežu i obojiti ona polja koja nisu slobodna.

Potrebna znanja: rekurzije, naredba FILL, interakcija sa zadanim funkcijama, liste