

Zadatak ZASTAVA	Autor: Frano Mihaljević
------------------------	--------------------------------

Zadatak rješavamo koristeći naredbu za pomicanje kornjače unaprijed (FD) te naredbe za okretanje kornjače za zadani kut (LT, RT). Rješavanje zadatke moguće je skratiti upotrebom naredbe REPEAT, iako njena upotreba nije nužna. Službeno rješenje tako ne sadrži tu naredbu i crta prvo donji lijevi pravokutnik, potom se pomiče i crta gornji lijevi pravokutnik pa gornji desni te konačno donji desni pravokutnik i spaja ga s donjim lijevim.

potrebno znanje: osnovne naredbe za pokretanje kornjače

Zadatak ILUZIJA	Autor: Marija Gegić
------------------------	----------------------------

Za crtanje lijevog dijela iluzije, potrebno je bilo nacrtati središnju kružnicu polumjera 30 piksela. Zatim je bilo potrebno nacrtati 5 manjih kružnica polumjera 15 piksela, pazeći na to da udaljenost središta manjih kružnica od središta središnje kružnice iznosi 55 piksela i da se za tu duljinu moramo pomaknuti bez ostavljanja traga. Za crtanje svake manje kružnice, možemo krenuti iz središta središnje kružnice, pomaknuti se za 55 piksela, nacrtati manju kružnicu, vratiti se u polazno središte i okrenuti za 72° . Na sličan način crtamo i desni dio iluzije, pazeći na to da je udaljenost središta središnjih kružnica 220 piksela i da je udaljenost središta većih kružnica od središnje desne kružnice 80 piksela.

potrebno znanje: osnovne naredbe za pomicanje kornjače

Zadatak UKRAS	Autor: Antea Hadviger
----------------------	------------------------------

Za rješavanje zadatka najbolje je koristiti petlju REPEAT koja šest puta ponavlja crtanje uzorka koji se sastoji od dvaju trokuta. U službenom rješenju, crtanje kreće na rubu šesterokuta sastavljenog od unutarnjih šest trokuta. Najprije se crta trokut unutar šesterokuta, a zatim vanjski trokut koji jednu od stranica dijeli s unutarnjim. Nakon pomicanja unaprijed i okreta udesno za 60° , ponavljamo istu radnju. Dakako, kako je broj trokuta konstantan, zadatak je moguće riješiti i bez korištenja petlje, ponavljajući isti niz naredbi šest puta.

potrebno znanje: osnovne naredbe za pomicanje kornjače

Zadatak CIJENA	Autor: Antea Hadviger
-----------------------	------------------------------

Primijetim najprije da je u zadatku potrebno napisati funkciju koja vraća isključivo jedan broj, a ne ispisuje dvije znamenke jednu za drugom. Ako znamo da moramo sastaviti dvoznamenkasti broj ab od znamenaka a i b , taj broj jednak je $10a+b$ jer je a broj desetica, a b broj jedinica.

Kako je navedeno u zadatku, za pločicu s brojem 6 ili 9 ne možemo okrenuti kako bi predstavljala jedan, odnosno drugi od tih brojeva. Zato svaku pločicu koja je zadana kao broj 9 okrećemo kako bi predstavljala broj 6 jer je potrebno ispisati najmanji broj koji je moguće dobiti pa ni u kojem slučaju nije povoljnije koristiti broj 9 umjesto 6. Ovo ne možemo primijeniti ni na koji drugi par znamenki.

Nadalje, potrebno je odrediti redoslijed znamenki. Kako želimo sastaviti što manji broj, uvijek je povoljnije da manja znamenka bude znamenka desetica, a veća znamenka jedinica. Pomoću te provjere odredimo želimo li vratiti $10a+b$ ili $10b+a$. Ipak, dolazi do problema ako je jedna od znamenaka jednaka nuli. Tada bi ovakav algoritam vratio jednoznamenkast broj jer bi nulu postavio na mjesto desetica, a iz teksta zadatka poznato je da je tražena cijena sigurno dvoznamenkasta. Zato je potrebno uvesti dodatnu provjeru koja u slučaju da je neka od znamenki jednaka nuli, tu znamenku postavlja na drugo mjesto, čime smo osigurali da ćemo vratiti dvoznamenkast broj.

potrebno znanje: naredba if

Zadatak DRVCA	Autor: Marija Gegić
----------------------	----------------------------

Za dobivanje 10% bodova na ovom zadatku, dovoljno je bilo nacrtati ravnu crtu duljine a i jedan trokut stranice b .

Za dobivanje svih bodova, potrebno je bilo uz pomoć neke petlje, primjerice FOR ili REPEAT, ponavljati crtanje svakog pojedinog drvca, pazeći na to da svaki put pamtimmo broj trokuta koje je potrebno nacrtati u krošnji i visinu debla drvca kojeg crtamo. U svakom koraku je potrebno broj trokuta u krošnji povećati za 1, te visinu debla povećati za x . To možemo napraviti uz pomoć naredbe MAKE. Pojedino drvec možemo nacrtati tako da nacrtamo pravokutnik širine x i pripadne visine koji čini deblo, te nad njime nacrtamo odgovarajući broj jednakokraničnih trokuta stranice b . Za implementacijske detalje pogledajte službeno rješenje.

potrebno znanje: petlje, naredba MAKE

Zadatak JOSIP	Autor: Frano Mihaljević
----------------------	--------------------------------

Promotrimo ovaj problem kada je n jednak nekoj potenciji broja 2. Potencija broja 2 je broj koji kad rastavimo na proste faktore, on sadrži samo faktor 2. Dakle, to su brojevi 2, 4, 8, 16, 32, ... Kada je n jednak nekom takvom broju, nakon svakog kruga brisanja brojeva, ostat će paran broj brojeva, osim nakon zadnjeg kruga kad ostaje samo jedan broj. Budući da će uvijek u krugu biti paran broj brojeva, uvijek ćemo obrisati posljednji broj u krugu, što znači da u idućem krugu nužno na ploči ostaje prvi broj - broj 1. Dakle, kada je n jednak nekoj potenciji broja 2, posljednji broj koji ostaje je broj 1.

Možemo primijetiti da će u krugu u nekom trenutku, nakon što obrišemo određen broj brojeva, ostati broj brojeva koji je jednak potenciji broja 2. Tada znamo da je sljedbenik onog broja koji smo posljednji obrisali rješenje zadatka.

Odnosno, ako nam je nakon x brisanja broj preostalih brojeva potencija broja dva, tada je rješenje broj $2^x + 1$ (2^x je broj koji posljednji obrišemo, a idući je rješenje - otuda ovih $+1$).

Tako u zadatku možemo samo naći najveću potenciju broja 2 koja je manja od broja n te pronaći x kao n minus ta potencija i ispisati rješenje $2^x + 1$.

Za 80% bodova bilo je dovoljno napraviti simulaciju - prepisivati nakon svakog kruga brisanja brojeve u drugu listu i vidjeti koji broj ostaje posljednji, no to je rješenje presporo u posljednja dva test podatka.

potrebno znanje: matematički pristup problemu, petlje

Zadatak ŽVRLJA	Autor: Ivan Paljak
-----------------------	---------------------------

Zadatak ŽVRLJA klasičan je primjer zadatka u kojem koristimo rekurzije, odnosno, funkcije koje unutar svoje definicije pozivaju same sebe. Zamislimo na trenutak da crtamo Mirkovu žvrlju nakon :k-tog koraka te da "magično" znamo nacrtati žvrlju nakon (:k-1) koraka. Nazovimo zasad tu magičnu funkciju STARA_ZVRLJA.

Zadatak bi tada bio lagan, a implementacija bi izgledala otprilike ovako:

```
to ZVRLJA :n :d :k :l
  repeat :n [
    STARA_ZVRLJA
    fd :d rt 360/:n
  ]
end
```

Problem je u tome što nemamo na raspolaganju "magičnu" funkciju STARA_ZVRLJA, ali pomaže nam činjenica što je izgled stare žvrlje gotovo identičan izgledu žvrlje nakon :k-tog koraka. Odnosno, ako funkcija ZVRLJA :n :d :k :l crta žvrlju nakon :k-tog koraka, onda funkcija ZVRLJA :n (:d/:l) (:k-1) :l crta žvrlju nakon (:k-1) koraka, a to je upravo funkcionalnost koju želimo od naše magične funkcije. Naš kod sada izgleda ovako:

```
to ZVRLJA :n :d :k :l
  repeat :n [
    ZVRLJA :n :d/:l :k - 1 :l
    fd :d rt 360/:n
  ]
end
```

Naizgled imamo rekurzivno rješenje koje zadovoljava uvjete zadatka, ali, pokrenete li ovaj program, vidjet ćete da se ništa na ekranu ne događa, a funkcija je završila u beskonačnoj petlji. Problem je u tome što je naše rekurzivno razmišljanje valjano samo do određene razine. Preciznije, smisao se gubi kada je :k jednak 0. Naime, mi tada tvrdimo da izgled žvrlje nakon 0-tog koraka ovisi o izgledu žvrlje nakon (-1)-og koraka što, dakako, nema smisla. Srećom, izgled žvrlje nakon 0-tog koraka nam je poznat i trivijalan, tada ništa nije nacrtano. Konačno, dodavanjem ovog zaustavnog uvjeta dobivamo kod:

```
to ZVRLJA :n :d :k :l
  if :k=0 [stop]
  repeat :n [
    ZVRLJA :n :d/:l :k - 1 :l
  ]
end
```

```

    fd :d rt 360/:n
  ]
end

```

Sada imamo rekurziju koja ispravno crta žvrlju i time smo osvojili 50% bodova na zadatku. Za preostalih 50% bodova bilo je potrebno izračunati od koliko se ukupno n -terokuta sastoji žvrlja. Sličnim razmišljanjem i blagim modifikacijama gore napisane funkcije taj broj možemo lako izračunati pa to ostavljamo čitatelju za vježbu. Taj je pristup i implementiran u službenom rješenju.

Ovdje ćemo predstaviti matematički pristup problemu za one koji žele znati više. Primijetimo da smo u prvom koraku nacrtali 1 n -terokut, u drugom koraku nacrtali smo n n -terokuta, u trećem koraku nacrtali smo n^2 n -terokuta te smo općenito u nekom koraku nacrtali n puta više n -terokuta nego u prethodnom (zato što smo u svakom vrhu n -terokuta iz prethodnog koraka crtali po jedan novi).

Nakon k koraka imamo ukupno $1 + n + n^2 + n^3 + \dots + n^{k-1}$ n -terokuta. Ovu formulu mogli bismo jednostavno izračunati pomoću petlji, no poigrajmo se još malo dobivenim izrazom. Označimo li naše rješenje sa S , dobivamo izraz:

$$S = 1 + n + n^2 + n^3 + \dots + n^{k-1}$$

Pomnožimo li svaku stranu jednadžbe sa $(1 - n)$ dobivamo:

$$\begin{aligned}
 (1 - n)S &= (1 - n)(1 + n + n^2 + n^3 + \dots + n^{k-1}) \\
 (1 - n)S &= 1 + n + n^2 + n^3 + \dots + n^{k-1} - n^2 - n^3 - \dots - n^{k-1} - n^k \\
 (1 - n)S &= 1 + (n - n) + (n^2 - n^2) + \dots + (n^{k-1} - n^{k-1}) - n^k \\
 (1 - n)S &= 1 - n^k \\
 S &= (1 - n^k) / (1 - n) = (n^k - 1) / (n - 1)
 \end{aligned}$$

potrebno znanje: Rekurzije

Zadatak SUDOKU	Autor: Mihael Liskij
-----------------------	-----------------------------

Zadatak SODOKU ima dva osnovna načina rješavanja: rješenje koristeći pravila sudoka i rješenje pomoću bruteforce-a.

Pomoću prvog načina se moglo lako riješiti slučajeve u kojima nedostaju najviše tri broja gledajući retke/stupce i traženjem broja koji jedini nedostaje u tom retku/stupcu. Kako je taj broj jedinstven samo ga ubacimo u matricu i ponavljamo postupak traženja broja koji nedostaje. Ovim postupkom se moglo osvojiti 70% bodova.

Za slučaj kada je nedostajalo četiri broja se moralo malo pametnije gledati jer se moglo dogoditi da ne postoji redak ili stupac u kojem nedostaje samo jedan element (ti slučajevi se mogu pronaći među zadnjim test primjerima). Zbog toga je umjesto gledanja koji broj nedostaje u nekom retku/stupcu bilo potrebno gledati gdje neki broj sigurno ne može biti i na jedino preostalo slobodno mjesto staviti taj broj. S tim rješenjem se moglo dobiti sve bodove.

Drugi način rješavanja je pomoću bruteforce-a, tj. isprobavanjem svih mogućnosti koje je dano kao službeno rješenje. Kako bi riješili zadatak prvo pišemo pomoćnu funkciju koja nam provjerava ako dana matrica poštuje sva pravila SUDOKA (jedinstveni brojevi u redcima, stupcima i grupama). Nakon toga je potrebno napisati rekurzivnu funkciju koja pronalazi prvi broj koji nedostaje, na njegovo mjesto redom pokušava ubaciti sve moguće brojeve (od 1 do 6) i poziva samu sebe ponovno. Kada funkcija ne uspije pronaći ni jedno mjesto na kojem broj nedostaje, poziva se pomoćna funkcija za provjeru točnosti i ako je sve dobro se matrica ispisuje i funkcija završava sa radom.

Oba rješenja su implementacijski dosta zahtjevnja pa se radi olakšavanja rada sa matricama preporuča istražiti mogućnosti ARRAY-a i MDARRAY-a u logu što je korišteno i u službenom rješenju.

potrebno znanje: Rad sa listama