

| | |
|------------------------|----------------------------|
| Zadatak PARKING | Autor: Marija Gegić |
|------------------------|----------------------------|

Jedan od mogućih načina rješavanja ovoga zadatka jest da prvo nacrtamo pravokutnik visine 10 i širine 200 piksela. Zatim se pomaknemo za 15 piksela udesno od njegovog donjeg lijevog ruba, te nacrtamo pravokutnik širine 10 i visine 200 piksela. Zatim se pomaknemo za dodatnih 125 piksela udesno i nacrtamo još jedan takav pravokutnik. Opisano rješenje donosilo je 50% bodova. Za preostalih 50%, potrebno je bilo nacrtati još dva manja pravokutnika visine 10 i širine 40 piksela, tako da se za 15 piksela udaljimo od donjeg desnog ruba pravokutnika visine 200 piksela i iz te pozicije započnemo crtanje manjeg pravokutnika.

Potrebna znanja: osnovne naredbe za pomicanje kornjače

| | |
|---------------------|------------------------------|
| Zadatak MLIN | Autor: Antea Hadviger |
|---------------------|------------------------------|

Crtež se sastoji od triju kvadrata i četiriju dužina. Sa skice znamo kolika je duljina stranice najvećeg kvadrata. Kako su linije između kvadrata dugačke 30 piksela, možemo izračunati da je duljina stranice srednjeg kvadrata jednaka $180-30-30=120$ piksela, a unutarnjeg $120-30-30=60$ piksela. Kvadrata možemo nacrtati tako da tri puta iskoristimo petlju REPEAT, pazeći da kornjača ne ostavlja neželjeni trag i da je točno pozicionirana za početak crtanja svakog kvadrata. Na kraju, za osvajanje svih bodova, potrebno je nacrtati četiri dužine, što je također moguće koristeći petlju REPEAT.

Potrebna znanja: osnovne naredbe za pomicanje kornjače

| | |
|------------------------|--------------------------------|
| Zadatak OBELISK | Autor: Frano Mihaljević |
|------------------------|--------------------------------|

Budući da se jednakokračan trapez kojeg je potrebno nacrtati sastoji od tri jednakokranična trokuta, lako možemo zaključiti da duljina njegove gornje stranice iznosi 100 piksela. Sada su nam poznate sve stranice i kutovi tog trapeza, pa ga možemo nacrtati korištenjem osnovnih naredbi za pomicanje kornjače. Rješenje koje je crtalo samo to donosilo je 25 bodova.

Da nacrtamo obelisk, potrebno se s polovišta gornje stranice trapeza pomaknuti za 240 piksela bez ostavljanja traga. Time smo se pomakli na polovište stranice jednakokraničnog trokuta koji se nalazi na vrhu obeliska. Nakon crtanja trokuta, potrebno se okrenuti za kut od $180^{\circ}-87^{\circ}$, odnosno za 93° koliko iznosi neoznačeni unutarnji kut većeg trapeza. Zatim se potrebno pomaknuti za 200 piksela te zapamtiti tu poziciju koristeći naredbe MAKE i POS. Time smo nacrtali jedan krak.

Nakon toga se vratimo za 200 piksela unazad te nacrtamo i drugi krak trapeza, nakon čega se korištenjem naredbe SETPOS pozicioniramo na zapamćeni donji vrh prethodno nacrtanog kraka i time nacrtamo donju osnovicu velikog trapeza.

Potrebna znanja: osnove koordinatne grafike

| | |
|---------------------|--------------------------------|
| Zadatak KULA | Autor: Frano Mihaljević |
|---------------------|--------------------------------|

Korisno je uočiti da se Mirkova kula od karata zapravo sastoji od jednakokračnih trokuta. U najnižem retku kule se nalazi točno n takvih trokuta (nacrtanih bez osnovice), a u svakom sljedećem retku se nalazi jedan trokut manje. Budući da znamo da se u najširem retku nalazi točno n trokuta, a znamo i da je širina tog retka d , lako možemo odrediti duljinu osnovice svakog trokuta tako da podijelimo širinu retka s brojem trokuta u retku. Duljina osnovice svakog trokuta je, dakle, d/n . Na jednak način zaključujemo da je visina svakog trokuta jednaka v/n . Sad kad znamo visinu i širinu svakog trokuta, korištenjem naredbi POS i SETPOS možemo vrlo jednostavno nacrtati svaki trokut: najprije nacrtamo osnovicu trokuta (osim ako je trokut u najširem retku) i zapamtimo njezin lijevi i desni kraj. Zatim se pomaknemo na polovište te osnovice, te se bez ostavljanja traga pomaknemo za v/n da dođemo u nasuprotni vrh trokuta. Zapamtimo poziciju vrha, te se redom pozicioniramo na lijevi kraj osnovice, pa ponovno na vrh, pa na desni kraj osnovice.

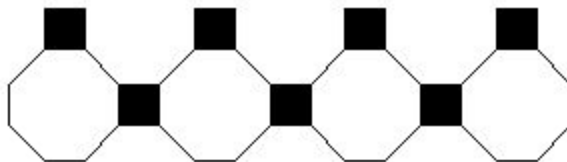
Ovaj postupak ponovimo odgovarajući broj puta da nacrtamo sve trokute u jednom retku, nakon čega se pomaknemo na poziciju iz koje počinjemo crtati novi redak i ponovimo isti postupak, pazeći da sada crtamo jedan trokut manje nego u prethodnom retku. Postupak ponavljamo sve dok je broj trokuta u retku veći ili jednak 1. Da ne bismo morali provjeravati prije crtanja svakog trokuta nalazi li se on u zadnjem retku, možemo sve trokute crtati tako da ne ostavljamo trag prilikom crtanja osnovice, a zatim iznad svakog nacrtanog retka nacrtati crtu koja spaja gornje vrhove prvog i zadnjeg trokuta u retku. Za implementacijske detalje proučite službeno rješenje.

Potrebna znanja: petlje, osnove koordinatne grafike

| | |
|------------------------|------------------------------|
| Zadatak KUHINJA | Autor: Antea Hadviger |
|------------------------|------------------------------|

Primijetimo najprije da se iznad svih osmerokutnih pločica, osim onih u najvišem retku, nalazi kvadratna pločica. Također se desno od svih osmerokutnih pločica, osim onih u zadnjem stupcu, nalazi kvadratna pločica. Korištenjem te opaske, rješavanje zadatka se svodi na crtanje n redaka koji se sastoje od m osmerokuta.

Iznad svakog osmerokuta u retku, kao i između svaka dva susjedna osmerokuta, treba se nalaziti kvadrat ispunjen crnom bojom. Primjer jednog takvog retka s 4 osmerokuta (koji odgovara prvom test primjeru iz zadatka) nalazi se na sljedećoj slici:



Nakon crtanja svakog retka, potrebno se pravilno pozicionirati prije početka crtanja sljedećeg retka, tako da se ispunjeni kvadrat iznad prvog osmerokuta nacrtanog u prošlom retku nalazi točno ispod prvog osmerokuta kojeg ćemo crtati u sljedećem retku. Za implementacijske detalje proučite službeno rješenje.

Potrebna znanja: petlje, IF, naredba FILL

| | |
|-----------------------|----------------------------|
| Zadatak POLICA | Autor: Marija Gegić |
|-----------------------|----------------------------|

Za osvajanje 10% bodova na ovom zadatku, dovoljno je bilo nacrtati praznu policu koja se sastoji od četiri pravokutnika i n pravilno raspoređenih crta.

U test primjerima vrijednim dodatnih 20% bodova, lista l se sastojala od samo jednog elementa, pa je dovoljno bilo na najnižoj pregradi nacrtati jednu knjigu visine h i širine zadane listom l .

U test primjerima vrijednim dodatnih 20% bodova, trebalo je nacrtati točno dvije knjige. One su mogle ili obje stati na jednu pregradu ili ih je bilo potrebno postaviti na prve dvije pregrade. Zato je prije crtanja bilo potrebno provjeriti je li zbroj njihovih širina manji ili jednak širini police. Ako jest, onda je trebalo nacrtati dva pravokutnika odgovarajućih širina jedan do drugog. Inače je trebalo svaki pravokutnik nacrtati na zasebnoj pregradi.

Za osvajanje svih bodova na ovom zadatku, trebalo je znati silazno sortirati listu. Jedan od mogućih načina da se dobije silazno sortirana lista jest korištenje funkcije SORT koja listu sortira uzlazno te funkcije REVERSE koja vraća listu u obrnutom poretку. Primijetimo da postavljanje trenutno najdeblje od preostalih knjiga možemo realizirati korištenjem naredbe BUTFIRST ili BF, a postavljanje trenutno najtanje od preostalih knjiga možemo realizirati korištenjem naredbe BUTLAST ili BL. Primjenom tih dviju naredbi ćemo dobiti listu bez prvog/zadnjeg elementa. Na samom početku crtanja, stvorit ćemo varijablu n nacrtano u kojoj ćemo pamtit zbroj širina svih knjiga koje smo do sada postavili, odnosno zbroj širina svih knjiga koje smo do sada nacrtali na trenutnoj pregradi. Zatim ćemo pokušati postaviti najdeblju

knjigu, tako da provjerimo je li zbroj prvog elementa liste :l i vrijednosti varijable :nacrtno manji ili jednak širini police. Ako jest, onda najdeblju knjigu možemo postaviti, pa povećamo varijablu :nacrtno, nacrtamo pravokutnik koji predstavlja tu knjigu i maknemo ju iz liste korištenjem naredbe BF i ispočetka ponovimo postupak. Kad više ne možemo staviti najdeblju knjigu, pokušat ćemo postaviti najtanju knjigu, tako da provjerimo je li zbroj zadnjeg elementa liste :l i vrijednosti varijable :nacrtno manji od širine police. Ako jest, onda najtanju knjigu možemo postaviti, pa povećamo varijablu :nacrtno, nacrtamo odgovarajući pravokutnik i maknemo zadnju knjigu iz liste korištenjem naredbe BL i ponovimo postupak. Kada više ne možemo postaviti trenutno najtanju knjigu, onda moramo početi postavljati knjige na novu pregradu. Vrijednost varijable :nacrtno zato postavimo na 0 te pomaknemo kornjaču na početak nove pregrade. Postupak ponavljamo dok nismo postavili sve knjige, odnosno dok god još ima elemenata u listi :l.

Potrebna znanja: liste, sortiranje

| | |
|------------------------|-----------------------------|
| Zadatak PUNOKUT | Autor: Mihael Liskij |
|------------------------|-----------------------------|

Zadatak je osmišljen po uzoru na klasične rekurzije, ali složen tako da se mora dobro paziti na to s koje strane mnogokuta se pozivaju.

U podzadatku s jednim elementom potrebno je samo nacrtati mnogokut s brojem stranica zadanim u listi.

Drugi podzadatak se može riješiti kao klasična rekurzija gdje imamo glavni program koji crta središnji mnogokut, a on onda poziva rekurziju na polovištima stranica koja pazi da ne pozove rekurziju na stranici koja je u prethodnom koraku bila zajednička.

Zadatak postaje kompliciran u trenutku kada uz mnogokute na polovištima stranica uvedemo mnogokute na vrhovima. U tom trenutku zadatak više nije rješiv običnom rekurzijom. S obzirom na to da postoji dovoljno velika razlika u mnogokutima, nameće se rješenje u kojem koristimo dvije rekurzije, jednu za stranice, a drugu za vrhove te da se one međusobno po potrebi pozivaju. Drugo rješenje je koristeći samo jednu rekurziju, ali onda moramo u dodatnoj varijabli pamtit i nalazi li se trenutni mnogokut nalazi na stranici ili vrhu prethodnog (u službenom rješenju to je varijabla :pom).

Neovisno o odabranom rješenju, potrebno je paziti na nekoliko stvari koje su zajedničke svim rekurzijama. Kada se koriste varijable unutar samih rekurzija, potrebno je umjesto MAKE koristiti LOCALMAKE kako bi one ostale sačuvane tijekom narednih poziva rekurzija. Prilikom crtanja rekurzije moramo paziti da se

vratimo na isto mjesto s kojeg smo počeli crtati. Dodatno, vezano uz zadatak, moramo paziti koliko elemenata ima trenutna podlista i na temelju toga odlučiti trebamo li crtati samo trenutni mnogokut, mnogokute na stranicama te mnogokute na stranicama i vrhovima.

Okvirno i nepotpuno rješenje koje koristi dvije rekurzije bi bilo:

```
to na_stranici :l :d :p
  ponovi (item 1 :l) puta:
    fd :d/2
    if (moram crtati na stranici i još nisam na zadnjoj stranici):
      na_stranici (item 2 :l) :d*:p :p
    fd :d/2
    if (moram crtati na vrhu i još nisam na zadnjem ili predznanjem vrhu):
      na_vrhu (item 3 :l) :d*:p :p
    rt 360/(item 1 :l)
end
```

```
to na_vrhu :l :d :p
  ponovi (item 1 :l) puta:
    fd :d/2
    if (moram crtati na stranici):
      na_stranici (item 2 :l) :d*:p :p
    fd :d/2
    if (moram crtati na vrhu i još nisam na zadnjem vrhu):
      na_vrhu (item 3 :l) :d*:p :p
    rt 360/(item 1 :l)
end
```

Službeno rješenje koristi samo jednu rekurziju te :pom označava o kojoj rekurziji se radi. 0 ako je to početni mnogokut, 1 ako je to mnogokut na stranici te 2 ako je mnogokut na vrhu. Zbog preglednosti i jednostavnosti se preporuča rješenje s dvije odvojene rekurzije.

Potrebna znanja: rekurzije, liste

| | |
|------------------------|---------------------------|
| Zadatak MALOKUT | Autor: Ivan Paljak |
|------------------------|---------------------------|

Promotrimo najprije rješenje koje je donosilo 60% bodova. S obzirom na to da svaki pravokutnik može biti obojen u jednu od četiri boje, ukupan broj različitih bojenja svih pravokutnika iznosi 4^n , pri čemu :n predstavlja ukupan broj

pravokutnika u listi `:l`. Ako svaku od četiri boje predstavimo jednim brojem od 1 do 4, sva moguća različita bojenja možemo generirati rekursivnim načinom:

```
to gen :bojenje :n
  if (count :bojenje) = :n [provjera :bojenje stop]
  for[i 1 4 1][gen (se :bojenje :i) :n]
end
```

Pozivom `gen [] (count :1)` generirat ćemo sva moguća različita bojenja, pri čemu će nam svako pojedino bojenje biti predstavljeno listom koja sadrži brojeve od 1 do 4. `:i`-ti broj u listi će nam predstavljati boju kojom želimo obojiti `:i`-ti pravokutnik.

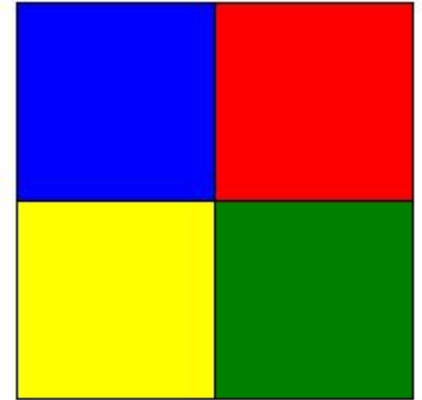
Uz to je još potrebno implementirati proceduru `provjeri :bojenje` koja će provjeriti valjanost bojenja zadanog listom `:bojenje`, tako da za svaka dva pravokutnika provjeri jesu li susjedni (odnosno postoji li barem jedna točka u njihovom presjeku), te ako jesu, razlikuju li se boje koje smo im dodijelili ovakvim bojenjem. Nađemo li tim postupkom na neka dva susjedna pravokutnika kojima smo dodijelili istu boju, znamo da ovakvo bojenje nije valjano. Ako takva dva pravokutnika ne postoje, onda ovakvo bojenje zadovoljava tražene uvjete iz zadatka, pa ga možemo nacrtati.

Ukupna složenost ovog rješenja je $:n^2 \cdot 4^n$.

Alternativno (ali također presporo) rješenje bi moglo korištenjem funkcije `RANDOM` svakom pravokutniku nasumično dodijeliti neku boju, provjeriti zadovoljava li takvo bojenje tražene uvjete iz zadatka te, ako ne zadovoljava, ponoviti postupak nasumičnog dodjeljivanja boja, sve dok ne nađe neko bojenje koje zadovoljava uvjete.

Uočimo pritom da u opisanom rješenju nismo nigdje koristili činjenicu da svi pravokutnici imaju neparne duljine stranica. Ako pravokutnik ima neparne duljine stranica, to znači da su mu i x-koordinate lijevog i desnog ruba i y-koordinate gornjeg i donjeg ruba različite parnosti. Uzmemo li neka dva pravokutnika koja dijele vertikalnu stranicu, sigurno će vrijediti da je x-koordinata donjeg lijevog ruba jednog od njih parna, a drugog neparna. Uzmemo li dva pravokutnika koja dijele horizontalnu stranicu, sigurno će vrijediti da je y-koordinata donjeg lijevog ruba jednog od njih neparna, a drugog parna. Ovakvo razmišljanje nas navodi na traženje rješenja koje će svakom pravokutniku dodijeliti neku od četiri boja u ovisnosti o parnosti njegovih koordinata donjeg lijevog vrha. Razlikujemo 4 slučaja: kad su obje koordinate parne, obje koordinate neparne, kad je x-koordinata parna a y-koordinata neparna te kad je x-koordinata neparna te y-koordinata parna.

S obzirom na to da su stranice svih pravokutnika paralelne s koordinatnim osima i pravokutnici se ne sijeku, najviše ih se četiri može zajedno dodirivati. Promotrimo slučaj u kojem četiri pravokutnika dijele jedan vrh, prikazan na slici desno (svi se ostali slučajevi mogu svesti ili na neki oblik ovog ili na slučaj u kojem je manje susjednih pravokutnika). Lako je odrediti parnost koordinata donjih lijevih vrhova pravokutnika ako znamo parnost koordinata središnjeg vrha, te time provjeriti da donji lijevi vrhovi ovih pravokutnika zapravo pokrivaju sva četiri prethodno navedena moguća slučaja. Želimo pronaći funkciju $f(x, y)$ koja će svakoj točki (x, y) pridružiti broj od 1 do 4, u ovisnosti o parnosti brojeva x i y ; taj će broj predstavljati boju u koju ćemo obojiti pravokutnik kojemu je donji lijevi vrh u točki (x, y) . Funkcija mora biti takva da svakom pravokutniku iz primjera desno pridružuje različit broj. Lako je provjeriti da je



$$f(x, y) = 1 + 2 * (|x| \bmod 2) + (|y| \bmod 2)$$

jedna od mogućih funkcija. S obzirom na to da navedena funkcija pridružuje različit broj svakom od 4 prethodno navedena slučaja parnosti koordinata, sigurni smo da će pridružiti različite boje susjednim pravokutnicima, jer znamo da će takvi pravokutnici imati različite parnosti bar jedne koordinate donjeg lijevog vrha.

Dakle, nakon što nacrtamo pojedini pravokutnik iz liste `:l`, oznaku boje kojom ćemo ga ispuniti odredimo kao `(1 + 2*(remainder (abs :x) 2) + (remainder (abs :y) 2))`, pri čemu su `:x` i `:y` koordinate njegovog donjeg lijevog vrha. Na kraju još preostaje taj pravokutnik ispuniti, tako da najprije koristeći naredbu `SETFC` postavimo boju ispune na onu koju smo odredili (neka nam brojevi 1, 2, 3, 4 primjerice označavaju redom crvenu, zelenu, plavu i žutu boju), te se zatim pozicioniramo unutar pravokutnika i ispunimo ga koristeći naredbu `FILL`.

Za one koji žele znati više: ovaj zadatak je specijalan slučaj problema kojim se bavi poznati teorem o četiri boje koji kaže da se svaka karta može obojiti s najviše četiri boje tako da su susjedna područja na karti obojena različito

Potrebna znanja: rekurzije, analiza problema