

Zadatke, test primjere i rješenja pripremili: Alen Rakipović <alen.rakipovic@gmail.com>, Goran Gašić <goran.gasic@gmail.com>, Ivo Sluganović <ivo.sluganovic@gmail.com>, Tomislav Gudlek <tgudlek@gmail.com> i Ante Đerek <ante.derek@gmail.com>.

Primjeri implementiranih rješenja su dani u priloženim izvornim kodovima koji nužno ne odgovaraju u svim detaljima ovdje opisanim algoritmima.

## **SPIRALA**

*Predložio: Ivo Sluganović*

*Potrebno znanje: jednodimenzionalna polja, osnovne petlje, kretanje u kvadratnoj mreži*

U zadatku je objašnjeno da je za ulazni broj  $N$  potrebno iscrtati spiralu veličine  $N$  redaka i  $N$  stupaca. Kako je  $N$  dovoljno malen, cijelu spiralu prvo ćemo zapisati u dvodimenzionalno polje, a zatim ćemo ispunjeno polje ispisati na ekran.

Postupak zapisivanja spirale možemo zamisliti kao obilazak kvadratne ploče dimenzija  $N \times N$  u kojem nam je cilj kretati se u istom smjeru sve dok u sljedećem koraku ne bismo izašli sa ploče ili stali na polje koje smo već posjetili. Kada se više ne možemo kretati u istom smjeru, okrećemo se ulijevo i nastavljamo kretanje. Na svako polje na koje se pomaknemo zapisujemo redni broj koraka (počevši od 1).

Kako bismo jednostavnije implementirali zamišljeno kretanje po ploči, možemo brojevima 0-3 definirati **4 smjera kretanja** (dolje, desno, gore, lijevo). Sada će nam okretanje ulijevo zapravo biti povećanje trenutnog smjera za 1 i eventualno cjelobrojno dijeljenje brojem 4.

Na početku se nalazimo na koordinati (0, 0), a u svakom koraku trenutnom retku i stupcu (x,y) dodajemo brojeve  $dx[smjer]$  i  $dy[smjer]$ , koji označavaju promjenu koordinate ukoliko napravimo pomak u trenutnom smjeru. Tako dobivamo (nx,ny), koordinate sljedećeg polja na koje bismo se pomaknuli ako zadržimo trenutni smjer. Ako se koordinate (nx,ny) nalaze izvan ploče ili na polju koje smo već posjetili i zapisali broj trenutnog koraka, mijenjamo trenutni smjer i ponovo računamo koordinate za sljedeći korak, te se pomičemo na koordinate (nx,ny). Kada konačno zapišemo broj  $N^2$ , stajemo s postupkom i ispisujemo cijelo polje.

## **MENI**

*Predložio: Ante Đerek*

*Potrebno znanje: dvodimenzionalna polja, stringovi, osnovne petlje*

Po opisu iz teksta zadatka, rečenice koje opisuju opcije potrebno je obrađivati u istom redoslijedu u kojem se pojavljuju na ulazu. To znači da nam je za određivanje svake od kratice potrebno znati koje smo kratice već iskoristili u prethodnim rečenicama. Nakon što smo pronašli prvo slovo koje još nije iskorišteno za kraticu, jednostavno je ispisati cijelu rečenicu s kraticom ograđenom uglatim zagradama. Skup iskorištenih slova možemo pamtiti nizom booleana veličine 26 ili na neki drugi način.

Razmotrimo načine na koje možemo obilaziti slova u redoslijedu opisanom u zadatku.

Za dovoljno vične u baratanju sa stringovima, izdvojiti sve riječi koje se pojavljuju u ulazu u dvodimenzionalno polje (svaka riječ u jedan redak) ne bi trebao biti problem. Nakon toga, dovoljno je proći matricom po stupcima krenuvši od prvog te testirajući za svako slovo na koje naletimo je li do sada iskorišteno za kraticu. U slučaju da nađemo neiskorišteno slovo, odmah prelazimo na ispis rečenice s ograđenom kraticom.

Moguće je i izbjeći korištenje dvodimenzionalnih polja tako da u jednodimenzionalnom polju zapamtimo početke svih riječi unutar rečenice te traženje po stupcima izvedemo s dvije ugniježdene petlje od kojih vanjska iterira po rednim brojevima slova u riječi (dakle, po stupcima), a unutarnja po riječima (dakle, njihovim počecima).

## LOZINKA

*Predložio: Tomislav Guldek*

*Potrebno znanje: stringovi, rekurzija, liste, sortiranje*

Svakom uzorku odgovara jedna ili više, ne nužno različitih, lozinki. Zadatak rješavamo tako da 'parsiramo' uzorak (rastavimo ga na logičke dijelove) pa, nakon toga, počevši od jednostavnih sastavnih dijelova uzorka prema većima generiramo niz svih mogućih različitih lozinki koje odgovaraju uzorcima. Na kraju, prebrojimo koliko različitih lozinki je u nizu koji odgovara početnom uzorku. Dakle, slutimo da se ovdje radi o nekakvom rekurzivnom postupku što nije čudno obzirom da je i sam pojam uzorka definiran rekurzivno.

Razmatrimo načine dopuštene kombinacije uzoraka, tj. dopuštenih načina da od manjih uzoraka napravimo veće. Neka su zadana dva uzorka  $a$  i  $b$  i neka uzorku  $a$  odgovara niz lozinki  $S_a$ , a uzorku  $b$  niz lozinki  $S_b$ .

1. Ako uzorke  $a$  i  $b$  spojimo dobivamo uzorak  $ab$ . Niz lozinki koje odgovaraju novom uzorku dobivamo tako da uzmemo sve parova stringova  $s_1$  iz  $S_a$  i  $s_2$  iz  $S_b$ , te ih spojimo.
2. Ako uzorke  $a$  i  $b$  stavimo u zagrade i odvojimo crtom dobivamo uzorak  $(a|b)$ . Niz lozinki koji odgovara novom uzorku je unija nizova  $S_a$  i  $S_b$ .

Najkompliciraniji dio zadatka je parsiranje zadanog uzorka. Primjetite da je ovo vrlo slično parsiranju odnosno evaluaciji aritmetičkih izraza - na primjer, zamjenite slova znamenkama, vertikalnu crtu znakom  $+$ , i dobili ste aritmetički izraz sa zbrajanjem i množenjem (u kojem se znak množenja ne pojavljuje ali je impliciran, npr  $2(3+5)$  znači  $2*(3+5)$ ). Baš kao i evaluacija aritmetičkih izraza i ovaj problem se može riješiti na mnogo različitih načina, ali je u svakom od njih lako moguće napraviti sitnu pogrešku.

U priloženom rješenju koristi se takozvani 'recursive descent' parser. On razmatra jedan po jedan znak uzorka redom od prvog prema zadnjem te u svakom koraku prema prvom znaku radi jedno od sljedećeg:

- Ako se radi o malom slovu ili otvorenoj zagradi treba spojiti koristeći pravilo 1 ono što smo do sada parsirali sa sljedećom logičkom cjelinom koja je ili niz malih slova ili cijeli jedan uzorak koji završava odgovarajućom zatvorenom zagradom.
- Ako se radi u malom slovu sljedeća logička cjelina je jednostavno niz uzastopnih malih slova koji tu počinje
- Ako se radi o otvorenoj zagradi, nju preskačemo i znamo da sada trebamo očekivati neki broj uzoraka odvojenih sa vertikalnom crtom.
  - *Rekurzivno* pozivamo istu proceduru da pročitamo prvi od tih uzoraka.
  - Ako smo došli do vertikalne crte, preskačemo je te *rekurzivno* pozivamo istu proceduru da pročitamo drugi od tih uzoraka. Spajamo rezultat sa onim iz prethodnog koraka koristeći pravilo 2.
  - Ponavljamo prethodni korak sve dok ne dođemo do zatvorene zagrade. Nakon toga preskačemo i krajnju odgovarajuću zatvorenu zagradu.
- Ako se ne radi o malom slovu ili otvorenoj zagradi (dakle radi se o kraju ulaza vertikalnoj crti ili zatvorenoj zagradi) onda smo gotovi sa trenutnim (pod) uzorkom.

**Za one koji žele više**

Koji je najveći mogući broj različitih lozinki ako je uzorak duljine **N**? Napravi program koji koristi dinamičko programiranje da izračuna traženi najveći broj lozinki. Matematičkim postupkom nađi formulu za traženi broj. Također, vidi isto poglavlje u zadatku Zaporka za prvu podskupinu.