

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.

## **BENFORD**

*Predložio: Ivo Sluganović*

*Potrebno znanje: jednodimenzionalna polja, cijelobrojno dijeljenje, osnovne petlje*

U zadatku je velikim dijelom objašnjen potreban postupak: izračunati koliko učitanih brojeva počinje znamenkom 1, koliko ih počinje znamenkom 2, i tako sve do znamenke 9. U sljedećem koraku potrebno je provjeriti vrijedi li  $C_1 \geq C_2 \geq \dots \geq C_9$ , i ispisati prikladan rezultat: 0 ako vrijedi nejednakost, a tri broja  $K$ ,  $C_K$  i  $C_{K+1}$ , ako ne vrijedi nejednakost, gdje je  $K$  najmanja znamenka za koju je  $C_K < C_{K+1}$ , tj. za koju je broj brojeva koji počinju znamenkom  $K$  manji od broja brojeva koji počinju znamenkom  $K+1$ . Brojeve  $C_K$  pamtimo u polju sa 10 elemenata, i kada pročitamo jedan broj sa ulaza odgovarajući element povećamo za jedan. Za provjeru nejednakosti tj. pronalazak traženog broja  $K$  potrebna je jedna for-petlja.

Kako bismo uspješno riješili zadatak, potrebno je još za svaki pojedini broj pronaći njegovu prvu znamenku. Primjetimo da se cjelobrojnim dijeljenjem bilo kojeg broja u dekadskom sustavu brojem 10 gubi njegova posljednja znamenka, pa je npr.  $123/10 = 12$  (i ostatak 3, kojeg zanemarujemo). Ukoliko broju  $X$  želimo odrediti prvu znamenku, dovoljno ga je cjelobrojno dijeliti brojem 10 sve dok broj  $X$  ne postane manji od 10, kada možemo biti sigurni da se sastoji samo od jedne, prve znamenke početnog broja.

### **Za one koji žele više**

Benfordov zakon općenito kaže da će postotak brojeva koji počinju znamenkom  $K$  biti otprilike  $\log_{10}(K+1) - \log_{10}(K)$  - što znači da će više od 30% brojeva počinjati sa znamenkom 1. Provjeri da li to vrijedi za veličine svih nepraznih datoteka na tvom računalu.

## **FORMULE**

*Predložio: Alen Rakipović*

*Potrebno znanje: dvodimenzionalna polja, stringovi, razlomci, sortiranje*

Zadatak rješavamo u dva koraka: u prvom koraku određujemo poziciju svake formule  $P_K$  - broj centimetara koje ta formula preći do kraja staze. Ako redove staze označimo sa brojevima 0, 1, ...,  $L-1$  onda je  $P_K$  jednak ( $L$  - oznaka prvog reda  $K$ -te trake koji sadrži znak 'o').  $K$ -ta traka odgovara stupcima  $4K+1$ , ...,  $4K+3$ , pa tako broj  $P_K$  nađemo tako da sa jednom for petljom provjeravamo izgled jednog po jednog reda u tim stupcima.

Kada smo našli brojeve  $P_K$  onda se za dvije formule jednostavno izračuna koja dolazi ranije - to je ona za koju je razlomak  $P_K / B_K$  manji, odnosno ona sa manjom oznakom ako su ti razlomci jednaki. U drugom koraku sortiramo oznake formula koristeći upravo navedeno pravilo. Jer je  $N$  mali, za samo sortiranje možemo odabrati i implementirati bilo koji algoritam (na primjer selection sort) ili koristiti jednu od funkcija za sortiranje iz standardnih biblioteka (u C-u i C++-u).

### **Za one koji žele više**

Napravi program koji učitava dvije različite pozicije između kojih je prošla jedna sekunda i određuje brzinu svake formule. Postavi web kameru usmjerenu na prometnicu i konfiguriranu da slika svaku jednu sekundu, napravi program koji analizira snimke i određuje brzine svih automobila.

## **ZAPORKA**

*Predložio: Alen Rakipović*  
*Potrebno znanje: polja, stringovi*

U zadatku je objašnjeno kako se dobiva niz zaporki iz uzorka: potrebno je naći otvorenu zagradu '(' i na podniz ispred nje nadodati sve moguće podnizove koji se nalaze između znakova '|', zatim je na svaki od nastalih stringova potrebno nadodati podniz koji se nalazi iza zatvorene zagrade ')'. Otvorenu i zatvorenu zagradu možemo jednostavno naći svaku pomoću jedne for petlje, a nakon toga možemo sa jednom petljom proći kroz sadržaj zagrada i rastaviti ga na nizove. Na ovaj način ćemo za svaki od N uzoraka dobiti niz od nekoliko zaporki koje mu odgovaraju.

Nakon toga je potrebno naći zaporku koja se pojavljuje u svim uzorcima. To je moguće napraviti na više načina, a jedan od njih je da, za svaku zaporku koja odgovara prvom uzorku, izbrojimo sva pojavljivanja te zaporku u nizovima koji odgovaraju ostalim uzorcima pri čemu je potrebno paziti na da pojavljivanje pribrojimo samo jedan puta po uzorku (ako se ona nalazi više puta kada generiramo sve moguće podnizove). Ukoliko se neka zaporka pojavljuje N-1 puta (N je broj uzoraka) - to je tražena zaporka.

### **Za one koji žele više**

Uzorci u ovom zadatku su specijalni slučaj takozvanih regularnih izraza - moćnih alata široke upotreba u računarstvu. Da bi od naših uzoraka dobili regularne izraze potrebno je dopustiti proizvoljno korištenje zagrada te dodati operator ponavljanja '\*' - regularni izraz '(u)\*' znači da možemo nula ili više puta ponoviti izraz u sadržan u zgradama. Napravi program koji rješava ovaj zadatak za proizvoljne regularne izraze.