

ŽUPANIJSKO NATJECANJE 2011.

OSNOVNE ŠKOLE BASIC/PASCAL/C/C++

OPISI RJEŠENJA ZADATAKA

Ovo su opisi algoritama kojima su se mogli riješiti navedeni zadaci. Direktne implementacije se nalaze u priloženim kodovima, uz napomenu da su službena rješenja kodirana u programskom jeziku Pascal, dok kodovi u programskom jeziku C++ mogu sadržavati neke alternativne ideje i načine rješavanja.

1. zadatak (5. razred)	TRENER	40/200 bodova
-------------------------------	---------------	----------------------

Prvo trebamo otkriti koliko je bodova na kraju testa imao nogometaš po imenu Lin. To ćemo dobiti tako da učitamo vrijednosti iz prvog reda ulaza te svaki od ta tri broja koji predstavljaju oznake rupe pretvorimo u odgovarajući broj bodova. To možemo postići tako da se za svaku oznaku rupe posebno pitamo koja je i ukupnom broju bodova dodamo odgovarajuću vrijednost. Npr., ako je oznaka rupe „1“, tada ukupan broj bodova povećamo za 4 boda, itd. Ali, možemo i uočiti da su oznake rupa i broj bodova povezani matematičkom vezom.

```
lin=0;
za i=1 do 3 radi
{
    ucitaj(L);
    ako je L<>0 tada
        lin:=lin+L*2+2;
};
```

Ponovimo ovaj postupak i za nogometaše s imenima Feng i Tao. Nakon što odredimo te tri vrijednosti, trebamo pronaći onu(e) koje su najveće. Pri tome trebamo paziti da otkrijemo sve one koje su maksimalne ali iste vrijednosti.

```
ako je (lin>=feng) i (lin>=tao) tada
    ispiši('LIN ',lin);

ako je (feng>=lin) i (feng>=tao) tada
    ispiši('FENG ',feng);

ako je (tao>=lin) i (tao>=feng) tada
    ispiši('TAO ',tao);
```

2./1. zadatak (5./6. razred)	TELEFON	70/40/200 bodova
-------------------------------------	----------------	-------------------------

Označimo sa „Z“ znamenku koju trenutno biramo, a s „R“ znamenku koju smo prethodno birali. „R“ je ustvari vrijednost koja se trenutno nalazi ispod rupe na plastičnoj pločici. Sada trebamo uočiti dvije situacije o kojima moramo voditi računa tijekom simulacije biranja znamenki zadanog telefonskog broja. To su:

1. kada je Z **veća ili jednaka** od R;
2. i kada je Z **manja** od R.

Opišimo detaljnije korake koje treba izvesti za pojedini od ova dva slučaja. Prvi slučaj, možemo podijeliti na nekoliko koraka.

- prvi korak; kotač treba napravi jedan pomak u smjeru suprotnome od kazaljke na satu;

- drugi korak; sada je vrijednost R-a za jedan manja, a ukupan broj pomaka za jedan veći;
- treći korak; trebamo napraviti $(Z-R)$ pomaka da bi doveli znamenku Z ispod plastike;
- četvrti korak; sada vrijednost R postaje Z.

Drugi slučaj, isto možemo podijeliti na nekoliko koraka.

- prvi korak; kotač treba napraviti $(R-Z)+1$ pomak u smjeru suprotnome od kazaljke na satu. Tako ćemo znamenku Z dovesti u područje prije rupe na plastičnoj pločici;
- drugi korak; sada je vrijednost R-a ustvari $Z-1$; (iako nam to sada nije ni potrebno)
- treći korak; napravimo jedan pomak u smjeru kazaljke na satu i dovedimo znamenku Z ispod rupe;
- četvrti korak; sada vrijednost R postaje Z.

I na kraju, pomak se mora povećati za trenutnu vrijednost R-a jer se kotač mora vratiti u početno stanje. Naravno, neki od ovih koraka se mogu udružiti te zajednički nakodirati.

Pseudo kod:

```

R=0; pomak=0; //uoči da R treba inicijalizirati na nulu
učitaj(n);
za i=1 do n radi
{
    učitaj(Z);
    ako je Z>=R tada
    {
        pomak=pomak+1;
        R=R-1;
        pomak=pomak+(Z-R);
        R=Z;
    }
    inače
    {
        pomak=pomak+(R-Z)+1;
        pomak=pomak+1;
        R=Z;
    }
};
pomak=pomak+R;

```

3./2. zadatak (5./6. razred)

LIFT

90/70/200 bodova

Da bi riješili ovaj zadatak trebamo osmisliti rješenje za dva problema. Prvi problem je kako simulirati kretanje lifta po katovima. To ćemo postići uvođenjem pomoćne varijable „pomak“ koja će imati vrijednost 1 ili -1 ovisno o smjeru kretanja lifta.

```

ako je (c=='D') then
    pomak=1
else
    pomak=-1;

```

Drugi problem je kako pamtit katove koje su osobe odabrale pri ulasku u lift tako da nam bude najlakše simulirati izlazak osoba iz lifta na određenom katu te određivati broj osoba u liftu.

Jedan način je da sve odabrane katove stavljamo u jedan niz. Izlazak iz lifta na pojedinom katu simuliramo tako da svako pojavljivanje tog kata u nizu zamijenimo s nulom. Broj osoba u liftu dobivamo tako da prebrojimo sve komponente u nizu koje su različite od nule.

Međutim, bolje rješenje je za svaki kat pamtiti koliko je osoba baš taj kat odabralo kao svoje polazno odredište. Ovaj način će nam omogućiti jednostavno rješavanje postavljenih problema.

Pseudo kod:

```
za k=1 do n radi
{
    ako je (kat==1) ili (kat==5) tada
        pomak=pomak*(-1); //mijenjamo smijer

    lift[kat]=0;

    učitaj(kuda);
    povećaj(lift[kuda]);

    s=0;
    za i=1 do 5 radi s=s+lift[i];
    ispiši(s);

    kat=kat+pomak;
};
```

3. zadatak (6. razred)	SATOVI	90/200 bodova
-------------------------------	---------------	----------------------

Zamislamo da smo odlučili sve satove postaviti da pokazuju na M minuta. Koliko nam ukupno pritisaka na tipke treba za to? Moramo obići sve satove i izračunati broj pritisaka na svakom satu. Ako sat i pokazuje M_i minuta, onda ga možemo namjestiti da pokazuje M minuta na 2 načina: priskupujući tipku '+' dovoljan broj puta ili pritiskujući tipku '-' dovoljan broj puta.

- Ako je $M_i = M$, onda ne trebamo pritisnuti niti jednu tipku.
- Ako je $M_i < M$, onda '+' treba pritisnuti $M - M_i$ puta, a '-' $M_i + (60 - M)$ puta – odabrat ćemo, naravno, manji od ta dva broja.
- Ako je $M_i > M$, onda '-' treba pritisnuti $M_i - M$ puta, a '+' $(60 - M_i) + M$ puta – ponovno odabiremo manji od tih brojeva.

Kako ćemo odlučiti koliki je M ? Jednostavno – pogledat ćemo svaki M od 0 do 59 i odabrati onaj za kojeg je ukupan broj pritisaka najmanji.

Pseudo kod:

```
za M = 0, 1, ..., 59 radi
    broj_pritisaka = 0
za i = 1, 2, ..., N radi
    ako je  $M_i < M$  onda
        broj_pritisaka=broj_pritisaka+manji_od_brojeva( $M - M_i, M_i + 60 - M$ )
    inače
        ako je  $M_i > M$  onda
            broj_pritisaka=broj_pritisaka+manji_od_brojeva( $M_i - M, 60 - M_i + M$ )
    ako je broj_pritisaka < najmanji_broj_pritisaka onda
        najmanji_broj_pritisaka = broj_pritisaka, najbolji_M = M
ispiši
    najbolji_M i najmanji_broj_pritisaka
```

1. zadatak (7. razred)**SNJEGULJICA****40/200 bodova**

Za riješiti ovaj zadatak je dovoljno prepoznati da su pozicija patuljka na rang listi novog branja, pomak tog patuljka i pozicija na rang listi iz prošlog branja u direktnoj matematičkoj vezi. Veza je iskazana formulom: $\text{stara_pozicija} = \text{nova_pozicija} + \text{pomak}$.

Sada još ostaje riješiti ispis stare rang liste, a to možemo koristeći niz stringova u koji smo na odgovarajućim pozicijama upisivali odgovarajuća imena patuljaka.

Pseudo kod:

```
za i=1 do 7 radi
{
    učitaj(ime_patuljka);
    učitaj(pomak);
    lista[i+pomak]=patuljak; // lista je niz stringova
};
za i=1 do 7 radi
    ispiši(lista[i]);
```

2. zadatak (7. razred)**BOND****70/200 bodova**

U test primjerima vrijednima 40% bodova, dovoljno je s četiri if petlje provjeriti da li se traženi izraz može kreirati korištenjem samo jednog od operatora. Npr., ako je $A+B+C+D+E=F$ tada ispiši(A,'+',B,'+',C,'+',D,'+',E,'=',F).

Za sve bodove, moramo složiti sve moguće izraze (njih 256) uz pomoć svih operatora koji se **moгу ponavljati**. Kako se operatori mogu ponavljati, ovim načinom pokrivamo i prvu metoda u kojoj se pojavljuje samo jedan operator. Kako zadatak garantira jedinstvenost rješenja, tada će samo jedan od ovih 256 kreiranih izraza biti istinit.

```
o='+-*/';
za i=1 do 4 radi
    za ii=1 do 4 radi
        za iii=1 do 4 radi
            za iiia=1 do 4 radi
                {
                    kreiraj_izraz(A,o[i],B,o[ii],C,o[iii],D,o[iiia],E);

                    ako je izraz=F tada
                        ispiši (A,o[i],B,o[ii],C,o[iii],D,o[iiia],E);
                };
```

3. zadatak (7. razred)**SEDMOSMJERKA****90/200 bodova**

Kako se u zadatku garantira da **neće** postojati mogućnost višestrukog odabira puta za traženje **sljedećeg** (ne i prvog) slova u riječi, tada je dovoljno samo osmisлити kretanje po slovima unutar tablice od prvog do posljednjeg slova u riječi. Naravno, potrebno je osmisлити i traženi način ispisa iz zadatka što možemo postići optimalnim korištenjem uvjeta zadatka. (vidi sedmosmjerka.pas)

1. zadatak (8. razred)**TESLA****40/200 bodova**

Ako iz ovog zadatka izbacimo prateću priču, zadatak se svodi na traženje maksimuma niza brojeva. Spomenuti niz brojeva su udaljenosti koje iskrice moraju prijeći do polja na kome se nalazi prst. Kako sve iskrice istovremeno kreću prema prstu, očito je da će „pokus“ trajati onoliko dugo koliko je potrebno da i zadnja (najdalja) iskrica dođe do prsta.

Postoje dvije vrste iskrica koje se nalaze na ploči, crvene i plave. Kako do polja na kome se nalazi prst putuju tražeći najkraći put, tada je dovoljno pronaći vezu između koordinata iskrice i prsta.

Za crvene iskrice koje se mogu kretati samo u smjerovima lijevo-desno i gore-dolje, očito je da je ta veza iskazana formulom $d = \text{abs}(\text{prstx} - Cx) + \text{abs}(\text{prsty} - Cy)$.

Za plave iskrice ta veza nije odmah očita, ali se raspisivanje dolazi do jednostavne veze oblika:

```
ako je abs(prstx-Px) < abs(prsty-Py) tada
    d = abs(prsty-Py)
inače
    d = abs(prstx-Px);
```

2. zadatak (8. razred)**DALJINSKI****70/200 bodova**

Uočimo da sve možemo računati u minutama, na primjer, 11:50 je zapravo $11 \cdot 60 + 50 = 710$ minuta. Dan ima $24 \cdot 60 = 1440$ minuta, pa sve satove treba namjestiti da pokazuju neki (isti) broj minuta između 0 i 1339.

Promotrimo prvo rješenje u slučaju kada nema daljinskog upravljača.

Zamislimo da smo odlučili sve satove postaviti da pokazuju na M minuta. Koliko nam ukupno pritisaka na tipke treba za to? Moramo obići sve satove i izračunati broj pritisaka na svakom satu. Ako sat i pokazuje M_i minuta, onda ga možemo namjestiti da pokazuje M minuta na 2 načina: pritiskujući tipku '+' dovoljan broj puta ili pritiskajući tipku '-' dovoljan broj puta.

- Ako je $M_i = M$, onda ne trebamo pritisnuti niti jednu tipku.
- Ako je $M_i < M$, onda '+' treba pritisnuti $M - M_i$ puta, a '-' $M_i + (1440 - M)$ puta – odabrat ćemo, naravno, manji od ta dva broja.
- Ako je $M_i > M$, onda '-' treba pritisnuti $M_i - M$ puta, a '+' $(1440 - M_i) + M$ puta – ponovno odabiremo manji od tih brojeva.

Kako ćemo odlučiti koliki je M ? Jednostavno – pogledat ćemo svaki M od 0 do 1339 i odabrati onaj za kojeg je ukupan broj pritisaka najmanji.

Promotrimo sada slučaj kada na neke satove djeluje daljinski upravljač. Posve je svejedno kojim redom pritiskamo tipke – da li prvo na daljinskom pa na satovima, ili obratno, ili pak posve izmiješano. Zato možemo zamisliti da smo odlučili daljinski dirati tek na kraju. U tom trenutku svi satovi na koje utječe daljinski moraju pokazivati neko isto vrijeme D , a svi satovi na koje ne utječe daljinski konačno vrijeme M . Tada još, kao u gornjem razmatranju, pritisnemo odgovarajući broj puta tipku '+' ili '-' na daljinskom kako bi baš svi satovi pokazivali M minuta. Broj D ćemo ponovno odabrati isprobavajući sve kombinacije.

Skica algoritma:

```
za M = 0, 1, ..., 1339 radi
    za D = 0, 1, ..., 1339 radi
        broj_pritisaka = 0
        za i = 1, 2, ..., N radi
```

```

    ako na sat i ne utječe daljinski
        x = broj pritisaka da se namjesti sat sa  $M_i$  na M
        minuta
    inače, ako na sat i utječe daljinski
        x = broj pritisaka da se namjesti sat sa  $M_i$  na D
        minuta
    broj_pritisaka = broj_pritisaka + x
    y = broj pritisaka da se namjesti sat sa D na M minuta
    broj_pritisaka = broj_pritisaka + y

    ako je broj_pritisaka < najmanji_broj_pritisaka onda
        najmanji_broj_pritisaka = broj_pritisaka, najbolji_M = M
    ispiši najbolji_M (u ispravnom formatu) i najmanji_broj_pritisaka

```

3. zadatak (8. razred)

OSMOSMJERKA

90/200 bodova

U primjerima vrijednima 70% bodova se garantira da **neće** postojati mogućnost višestrukog odabira puta za traženje **sljedećeg** (ne i prvog) slova u riječi. Zato je dovoljno samo osmisliti kretanje po slovima unutar tablice od prvog do posljednjeg slova u riječi.

Za preostalih 30% test primjera, ipak je potrebno kodirati rekurziju koja će tražiti pravi put između nekoliko ponuđenih.