2.

Rješavanje problema programiranjem

2.5. Problemski zadaci

Rješavanje problema je traženje puta do cilja koji nije odmah dokučiv. Ako postupak koji će dovesti do rješenja opišemo algoritamskim jezikom, možemo upotrijebiti računalo i do rješenja doći brže. No put moramo pronaći *sami*.

Suočimo li se s novom problemskom situacijom obično se pojavi pitanje: "Odakle početi?" Dobro je slijediti faze koje je, imajući na umu matematičke zadatke, predložio američki profesor Georg Polya[1], a te su:

1. **Razumijevanje zadatka**
   (Što je zadano? Što je nepoznato? Kako glasi uvjet?)
   Tek kad pouzdano znamo odgovore na sva tri pitanja smijemo se upustiti u "avanturu" rješavanja problema. Dakle, moramo znati što tražimo, koji podaci su nam na raspolaganju te koja su eventualna ograničenja na koja treba paziti.
2. **Stvaranje plana**
   (Odnosi se na traženje veze između zadanog i nepoznatog, na prethodna iskustva; pomoći će sličan problem koji smo već prije riješili ili će se morati razmatrati pomoćni zadaci koji će dovesti do potpunog ili djelomičnog rješenja.)
   Nakon jasno iskazane veze između poznatog i nepoznatog pokušavamo se prisjetiti na koji smo se način "snalazili" u sličnim situacijama. Je li nam već na početku poznat postupak kako do rješenja doći ili se njemu tek trebamo domisliti? Polya kaže: "Stvoriti plan, doći do ideje rješenja – nije lako. Da bismo u tom uspjeli, treba nam mnogo toga. Potrebna su ranije stečena znanja, disciplina duha, koncentracija na cilj i još nešto: sreća. Izvršiti plan – mnogo je lakše. Za to treba uglavnom samo strpljenja."
   Ukoliko je moguće, postupak rješavanja problema možemo odmah opisati strukturama pseudojezika. Ukoliko je problem teži, koristimo govorni jezik i postepeno se "približavamo" strukturama pseudokoda.
3. **Izvršenje plana**
   Koristimo li računalo tada izvršenje plana podrazumijeva kodiranje, prevođenje i izvođenje odgovarajućeg programa. Možemo reći da je plan u načelu izvršen ako je algoritam rješenja zapisan u pseudokodu.
4. **Provjera dobivenog rješenja**
   (Može li se rezultat kontrolirati? Može li se on uočiti na prvi pogled?)
   U ovoj je fazi predviđen osvrt na algoritam rješenja: preispitivanje rezultata kao i puta koji je do tih rezultata doveo. Za neke karakteristične ulazne podatke (test primjere) provjeravamo hoće li algoritam dati očekivani rezultat. I pored zadovoljavajućih rezultata ovih provjera, ne možemo biti sigurni da će

[1] Polya, George, *Kako ću riješiti matematički zadatak*, 2. izdanje, Školska knjiga, Zagreb, 1966.

algoritam uvijek dati ispravno rješenje. Ipak, na ovaj se način mogu ispraviti mnoge greške.[2]

U zadacima koji slijede, uz algoritam rješenja zapisan u pseudojeziku, bit će opisane i faze koje su rješenju prethodile, te provjera danoga rješenja. Pored toga vidjet ćemo da se drukčijim pristupom ponekad može ostvariti bolje, efikasnije rješenje.

Zadaci 2.5.

Zadatak 1.

Banana srednje veličine ima oko 34 g ugljikohidrata. Koliko ćemo ugljikohidrata unijeti u organizam pojedemo li n takvih banana? Napišite program u pseudojeziku koji daje odgovor na postavljeno pitanje. Podatak n čita se s tipkovnice. Rezultat izrazite u dekagramima i gramima.

Odgovor:

1 dekagram (dag) = 10 g

Kako 1 banana ima 34 g ugljikohidrata, to n banana ima $(34 \cdot n)$ g ugljikohidrata. Primjerice za $n = 4$, količina ugljikohidrata je: 136 g = 13 dag 6 g.

Algoritam rješenja:

```
ulaz (n);
u := 34 * n;
izlaz ('Količina ugljikohidrata je ', u div 10 ,
       'dag ', u mod 10, ' g.');
```

Zadatak 2.

Proizvođač planira proizvesti najviše n litara ulja. Koliko petlitarskih posuda treba pripremiti za spremanje te količine ulja? Korištenjem pseudojezika napišite program koji daje odgovor na postavljeno pitanje. Podatak n unosi se pomoću tipkovnice.

[2] Algoritam je sigurno ispravan ukoliko za svaki ulaz daje ispravan rezultat. Testiranje samo nekih vrijednosti ulaznih varijabla nije garancija potpune ispravnosti algoritma. No pogreške koje se na ovaj način otkriju siguran su znak da je u postupku rješavanja napravljena logička greška i/ ili da rješenje nije cjelovito.

Odgovor:

Cjelobrojnim dijeljenjem `n div 5` doznajemo koliko bi pet litarskih posuda do vrha bilo napunjeno uljem, ostvari li se planirana proizvodnja. Ostatak cjelobrojnog dijeljenja `n mod 5` daje informaciju:

— je li to bilo sve ulje koje je trebalo spremiti (`n mod 5 = 0`);

— ili je ostalo još ulja koje treba spremiti ((`n mod 5`) $\neq 0$). U tom je slučaju poznato da je količina preostalog ulja manja od 5 litara.

Primjerice, za $n = 20$, `n div 5 = 4, n mod 5 = 0`; sve ulje se može spremiti u 4 pet litarske posude.

Za $n = 22$, `n div 5 = 4, n mod 5 = 2`; potrebno je 5 posuda: četiri će biti pune do vrha, a u petoj će biti dvije litre ulja.

Algoritam rješenja:

```
ulaz (n);
broj := n div 5;
ako je n mod 5 <> 0 onda broj := broj + 1;
izlaz ('Potreban broja posuda:  ', broj);
```

Zadatak 3.

Vrijeme potrebno da Zemlja "zaokruži" svoj put oko Sunca traje točno 365.2424 dana. Kažemo da je to jedna sunčeva ili astrološka godina. Usklađivanje kalendarske i astrološke godine prema gregorijanskom je izračunavanju vremena riješeno uvođenjem prijestupne godine. Tako obična godina ima 365 dana, a svaka prijestupna ima jedan dan više. Prijestupna godina je ona koja je djeljiva sa četiri. Iznimka su godine na prijelazu stoljeća: nije prijestupna ona koja je djeljiva sa 100, ali jest ona koja je djeljiva sa 400.

U pseudojeziku napišite program koji će ispisati je li godina g prijestupna ili nije. Pretpostavka je da se podatak g čita s tipkovnice i da pri učitavanju nije napravljena greška.

Odgovor:

Da bi godina bila prijestupna mora biti ispunjen barem jedan od uvjeta:

a) g je djeljivo s 4, ali nije djeljivo sa 100;

b) g je djeljivo s 400.

Uočimo tri izjave: A = "g je djeljivo s 4";
B = "g nije djeljivo sa 100";
C = "g je djeljivo s 400".

Uz ovaj zapis možemo reći da je godina prijestupna ako logički izraz (A I B) ILI C ima vrijednost *istina*. Zagrade u izrazu smijemo izostaviti budući da je operacija konjunkcije višeg prioriteta od operacije disjunkcije.

Algoritam rješenja:

```
ulaz (g);
ako je (g mod 4 = 0) I (g mod 100 <> 0)
                ILI (g mod 400 = 0) onda
     izlaz ('Godina je prijestupna.')
inače
     izlaz ('Godina nije prijestupna.');
```

Zadatak 4.

Prema redu vožnje autobus svakih m minuta ponovno kreće s polazne stanice. Za koliko se minuta autobus očekuje u polaznoj stanici ako nije bilo nepredviđenih zastoja, a prošlo je n sati od početka radnog vremena?
Napišite program korištenjem pseudojezika za zadani problem. Podaci za m i n unose se pomoću tipkovnice.

Odgovor:

Prije bilo kakvog računa treba uskladiti mjerne jedinice. Ako odlučimo vrijednosti iskazati u minutama, onda nas zanima gdje se autobus nalazi nakon $(60 \cdot n)$ minuta vožnje. Cjelobrojnim dijeljenjem `(60*n) div m` doznajemo koliko je puta u tom vremenu autobus odvozio cijelu rutu. Nakon toga možda je odvozio još jedan dio rute. Koliko mu je za to trebalo vremena dobijemo iz sljedećeg izraza: `(60*n) mod m`. No tada će mu do dolaska u polaznu stanicu trebati još `m-(60*n) mod m` minuta.

Primjerice, ako autobus svake 52 minute ponovno kreće s polazišta ($m = 52$), nakon 3 sata ($n = 3$) on će se, budući da je, `(60*3) div 52 = 3`, tri puta vratiti u polaznu točku. Potom će krenuti na četvrtu rutu i nakon tri sata bit će od polazne točke udaljen `(60*3) mod 52 = 24` minuta vožnje. Dakle, u polaznu će se stanicu vratiti za $52 - 24 = 28$ minuta.

Algoritam rješenja:

```
ulaz (m,n);
ako je (60*n) mod m = 0 onda
     izlaz ('Autobus je u polaznoj stanici.')
inače
     {
          d := m - (60*n) mod m;
          izlaz ('Autobus se u polaznoj stanici
                  očekuje za ', d,' minuta.')
     }
```

Zadatak 5.

Tražeći parking Ivo je skrenuo u Ulicu brijestova. Zapazio je da se njemu slijeve strane ispred svake kuće nalaze dva parkirna mjesta. Na toj su strani ulice kućni brojevi neparni. Na desnoj "parnoj" strani svaka kuća ima samo jedno parkirno mjesto. Ivo je prošao cijelu ulicu, ali nije imao sreće – sva su parkirna mjesta bila zauzeta. Napišite program u pseudojeziku koji će za učitani broj n koji je jednak najvećemu kućnom broju u Ulici brijestova, ispisati ukupan broj parkirnih mjesta u toj ulici.

Odgovor:

Na" parnoj" strani ulice ima: `n div 2 kuća; n div 2` parkirnih mjesta.
Na "neparnoj" strani ima: `n div 2 + n mod 2` kuća, dakle `(n div 2 + n mod 2) * 2` parkirnih mjesta.

Primjerice, za $n = 30$ ukupan je broj parkirnih mjesta jednak 45, a za $n = 31$ njihov je broj 47.

Algoritam rješenja:

```
ulaz (n);
p := 3 * (n div 2);
ako je (n mod 2) = 1 onda p := p + 2;
izlaz ('Broj parkirnih mjesta u Ulici brijestova je:  ', p);
```

Zadatak 6.

Slova treba zamijeniti znamenkama, svako slovo je jedna znamenka, ali tako da dijeljenje ima smisla: $KIM : A = MIA$.

U pseudojeziku napišite program koji provjerava jesu li unesene znamenke K, I, A i M rješenje danoga rebusa. (Napomena: znamenke se čitaju s tipkovnice. Pretpostavka je da pri unosu nije napravljena greška: svakom je slovu pridijeljena jedna od vrijednosti $0, 1, 2, \ldots, 9$; različitim su slovima pridijeljene različite vrijednosti.)

Odgovor:

Za unesene vrijednosti treba provjeriti jednakost: $KIM : A = MIA$. Na primjer, za $K = 1, I = 2, M = 7, A = 3$, dobivamo $127 : 3 = 723$, što očito nije rješenje.
Umjesto jednakosti $KIM : A = MIA$ jednostavnije je provjeriti: $KIM = MIA \cdot A$, $A \neq 0$. Uz pretpostavku da pri unosu nije napravljena greška, provjera $A \neq 0$ može se izostaviti. Budući da je za $A = 0$ lijeva strana jednakosti $KIM = MIA \cdot A$ različita od nule, a desna jednaka nuli, to niti jedna kombinacija kod koje $A = 0$ neće izjednačiti $KIM = MIA \cdot A$.

Algoritam rješenja:

```
ulaz (K,I,A,M);
ako je 100*K+10*I+M= (100*M+10*I+A)*A onda
```

```
        izlaz ('Ova kombinacija daje rješenje.')
inače
        izlaz ('Ova kombinacija ne daje rješenje.');
```

Zadatak 7.

Pustimo li tijelo da slobodno pada, ono u prvoj sekundi prevali 4.905 m, a u svakoj sljedećoj 9.81 m više nego u prethodnoj. Korištenjem pseudojezika napišite program koji računa koliki će put prevaliti tijelo za s sekundi. Podatak s čita se s tipkovnice. (Napomena: s je pozitivan cijeli broj; pretpostavimo da pri unosu nije napravljena greška.)

Odgovor:

Pređeni put inicijalno stavimo na 4.905 (`p_ukupno := 4.905`). Ta će se vrijednost ispisati ukoliko korisnika zanima koliki će put tijelo prevaliti za jednu sekundu ($s = 1$). Ukoliko je $s > 1$, tada za svaku "dodatnu" sekundu ukupnom putu treba pribrojiti put koji tijelo pređe u toj sekundi. Koliko dodatnih sekundi ima? Odgovor je $s - 1$. Dakle, operaciju pribrajanja treba ponoviti $s - 1$ puta. To se može ostvariti jednom od tri petlje (petlja `za`, petlja `dok je` ili petlja `ponavljati ... do`). Jedini problem je taj što vrijednost koju treba pribrojiti nije uvijek ista. Naime, tijelo u prvoj sekundi pređe put od 4.905 m, u drugoj 4.905 + 9.81 m, u trećoj (4.905 + 9.81) + 9.81 m i tako dalje. Stoga je nužno koristiti još jednu varijablu koja će u svakom koraku petlje ažurirati vrijednost koju treba pribrojiti putu. Neka to bude varijabla `p`. Njezina je inicijalna vrijednost 4.905. U svakom će se koraku petlje ta vrijednost uvećati za 9.81 (`p := p + 9.81`).

Ovisno o petlji tri su inačice algoritma:

a)
```
ulaz (s);
p := 4.905;
p_ukupno := p;
ako je s > 1 onda
      za i:= 1 to s-1 činiti
      {
            p := p + 9.81;
            p_ukupno := p_ukupno +p;
      }
izlaz ('Tijelo prevali ', p_ukupno, ' metara.');
```

b)
```
ulaz (s);
p := 4.905;
p_ukupno := p;
ako je s > 1 onda
```

```
        dok je s > 1 činiti
        {
            s := s - 1;
            p := p + 9.81;
            p_ukupno := p_ukupno +p;
        }
    izlaz ('Tijelo prevali ', p_ukupno, ' metara.');
```

c)

```
    ulaz (s);
    p := 4.905;
    p_ukupno := p;
    ako je s > 1 onda
        ponavljati
            s := s - 1;
            p := p + 9.81;
            p_ukupno := p_ukupno +p;
        do s = 1;
    izlaz ('Tijelo prevali ', p_ukupno, ' metara.');
```

Svako rješenje, pa i algoritamsko, dobro je provjeriti. Odaberemo takve ulazne podatke za koje je rješenje unaprijed poznato ili se do njega može doći jednostavnim računom bez pomoći računala. Potom provjerimo hoće li za te podatke i računalo "vođeno" danim algoritmom dobiti isti rezultat: nacrtamo "kućice" varijabli i pratimo kako se njihove vrijednosti mijenjaju izvođenjem naredbi algoritma. Primjerice, za $s = 1$, očekujemo da ukupna vrijednost prijeđenog puta bude 4.905, a za $s = 3$ rezultat bi trebao biti 44.145. Naime, označimo li sa p_1, p_2 i p_3 put tijela u prvoj, drugoj i trećoj sekundi, tada je

$$\begin{aligned} p_1 &= 4.905 \\ p_2 &= p_1 + 9.81 = 14.715 \\ p_3 &= p_2 + 9.81 = 24.525 \end{aligned}$$

a ukupan je put jednak

$$p_1 + p_2 + p_3 = 4.905 + 14.715 + 24.525 = 44.145.$$

Preostaje provjeriti hoće li i predloženi algoritmi dati isti rezultat. Primjerice, algoritam a) računalo će protumačiti na sljedeći način:

— za ulaz `s = 1` vrijednosti varijabla `p` i `p_ukupno` se inicijaliziraju na 4.905. Naredba grananja će za `s = 1` biti preskočena, stoga će to ostati i njihove konačne vrijednosti.

— za `s = 3` varijable se mijenjaju na sljedeći način:

`s`	3
`p`	24.525, ~~14.715~~, ~~4.905~~
`p_ukupno`	44.145, ~~19.62~~, ~~4.905~~
`i`	2, ~~1~~

Ispisat će se broj 44.145. Na sličan način provjerimo ispravnost algoritama b) i c).

Na ovom bi se zadatku valjalo još malo zadržati. Naime, predloženi algoritam rješenja[3] nije jedini. Promotrimo niz brojeva $p_1, p_2, p_3, \ldots, p_n, \ldots$, gdje je p_i put koji tijelo prevali u i-toj sekundi. Taj je niz aritmetički budući da je razlika svakoga člana i člana ispred njega jednaka $d = 9.81$. Kako je zbroj prvih n članova aritmetičkoga niza jednak:

$$S_n = (a_1 + a_n) = na_1 + \frac{n(n-1)}{2} \cdot d,$$

gdje je a_1 prvi član, a d razlika (diferencija) niza, to je put koji tijelo prevali za s sekundi jednak:

$$p_1 + p_2 + \ldots + p_s = 4.905 \cdot s + \frac{s(s-1)}{2} \cdot 9.81.$$

Posljednja jednakost vodi na sljedeći algoritam:

```
ulaz (s);
p_ukupno := 4.905*s + (s*(s-1)/2) * 9.81;
izlaz ('Tijelo prevali ', p_ukupno, ' metara.');
```

Primijetimo da je ovaj algoritam efikasniji od prethodnoga: iziskuje tri množenja, jedno dijeljenje, jedno zbrajanje i jedno oduzimanje za izračun cijeloga puta neovisno o tome koliki je s.

Zadatak 8.

Marta je napisala niz: $1, 2, -3, -4, 5, 6, -7, -8, 9, 10, \ldots$. Napišite program u pseudojeziku koji će izračunati zbroj prvih n članova Martina niza. Broj n čita se s tipkovnice.

Odgovor:

Martin niz dobijemo tako da popišemo prirodne brojeve redom, a potom nekima od njih promijenimo predznak. Predznaci se mijenjaju prema sljedećem pravilu: prva

[3] Algoritmi a), b) i c) razlikuju se samo u izboru petlje pa ih stoga smatramo istim rješenjem danoga problema.

četiri broja $+\ +\ -\ -$; svaka naredna četvorka opet $+\ +\ -\ -$. Na temelju ovih razmatranja možemo odrediti bilo koji član Martina niza.

Vrijednost zbroja z dobijemo tako da n puta:

— odredimo odgovarajući član niza;

— pribrojimo ga vrijednosti varijable z.

Naravno, početna vrijednost varijable z treba biti 0.

Algoritam rješenja:

```
ulaz (n);
z := 0;
za i := 1 to n činiti
{
     ako je (i mod 4 = 1) ILI (i mod 4 = 2) onda
          p := 1
     inače
          p := -1;
     z := z + p * i;
}
izlaz ('Zbroj:  ', z);
```

Provjerimo za $n = 7$: $1 + 2 - 3 - 4 + 5 + 6 - 7 = 0$. Predloženim algoritmom dobijemo isti rezultat:

n	7
i	7, ~~6~~, ~~5~~, ~~4~~, ~~3~~, ~~2~~, 1
p	−1, 1, 1, ~~−1~~, ~~−1~~, 1, 1
z	0, ~~7~~, 1, ~~−4~~, ~~0~~, 3, 1

Zadatak 9.

Pismenom ispitu pristupilo je n učenika. Oni su na testu mogli ostvariti najviše m bodova. Za prolaznu je ocjenu trebalo točno riješiti 50% testa. Napišite program u pseudojeziku koji će za svakog učenika unositi ostvarene bodove te ispisati postotak prolaznosti testa. Podaci n, m te bodovi koje su učenici na testu ostvarili, čitaju se s tipkovnice.

Odgovor:

Za prolaznu ocjenu učenik treba prikupiti najmanje $(0.5 \cdot m)$ bodova. Koliko je njih uspjelo prikupiti te bodove, dobit ćemo tako da za svakog od n učenika:

- učitamo ostvareni broj bodova;
- provjerimo je li taj broj dovoljan za prolaz, i ako jest, onda brojač "uspješnih" učenika povećamo za 1 (`broj := broj+1`).

Prije početka brojenja treba osigurati da je vrijednost ovoga brojača jednaka 0 (`broj := 0`).

Preostaje izračunati omjer (`broj/n`) i prikazati ga u postocima: (`broj/n`) · 100%.

Algoritam rješenja:

```
ulaz (n);
ulaz (m);
m := 0.5*m;
broj := 0;
za i := 1 do n činiti
     {
          ulaz (b) ;
          ako je b >= m onda broj := broj +1;
     {
p := (broj/n)*100;
izlaz ('Postotak prolaznosti je:  ', p,' %.');
```

Neka je pismenim ispitu pristupilo pet učenika i neka su oni od 100 mogućih, ostvarili redom: 52, 78, 25, 30 i 90 bodova. Ove ulazne podatke algoritam će "interpretirati" na sljedeći način:

n	5
m	50, ~~100~~
b	90, ~~30~~, ~~25~~, ~~78~~, ~~52~~
broj	3, ~~2~~, ~~1~~, ~~0~~
i	5, ~~4~~, ~~3~~, ~~2~~, ~~1~~
p	60

Postotak prolaznosti je 60%.

Zadatak 10.

Na pismenom ispitu bilo je moguće ostvariti najviše 70 bodova. Ispitu je pristupilo n učenika. Napišite program u pseudojeziku koji će za svakog učenika unositi ostvarene bodove, ispisati koji je najveći ostvareni broj bodova te koliko je učenika uspjelo postići taj rezultat.
Broj učenika n i brojevi bodova koje su ti učenici ostvarili učitavaju se s tipkovnice. Pretpostavlja se da pri učitavanju nije napravljena greška.

Odgovor:

Bodovi učenika čitaju se redom: najprije bodovi prvog, pa drugog i naposljetku n-tog učenika. Nakon svakog se učitavanja odredi koji je najveći broj bodova do tada učitan i taj se broj zapamti u varijabli `max`. Kada svi podaci budu učitani u varijabli `max` nalazit će se najveći broj bodova ostvaren na ispitu. Preciznije:

- Učitaju se bodovi prvog učenika i pridruže se varijabli `max`[4];
- Učitaju se bodovi drugog učenika. Ukoliko su ti bodovi veći od `max`, znači da je drugi učenik bio uspješniji od prvoga. Zato se u varijablu `max` zapišu bodovi drugog učenika.

Ovaj se postupak ponavlja: nakon svakog se učitanog broja bodova provjeri treba li vrijednost zapisanu u `max` zadržati ili zamijeniti novom većom vrijednošću.

Opisani postupak zapisan u pseudojeziku glasi:

```
ulaz (n);
ulaz (b);
max := b;
za i := 2 do n činiti
     {
          ulaz (b) ;
          ako je b > max onda max := b;
     }
izlaz ('Najveći broj bodova je:  ', max);
```

Da bi postavljeni zadatak u potpunosti bio riješen treba prebrojiti koliko je učenika ostvarilo maksimalni broj bodova. Uočimo trenutak u kojem varijabla `max` mijenja vrijednost. U tom je trenutku pronađen učenik čiji je broj bodova (`max`) veći od svih do tada učitanih bodova. No tada je to i jedini učenik koji ima toliko bodova pa stavimo da je ukupan broj učenika koji su postigli `max` bodova jednak 1 (`broj := 1`). Ako neko od sljedećih učitavanja otkrije da postoji učenik s istim brojem bodova, ovaj se brojač poveća za 1 (`broj := broj + 1`).

[4] Zapisano u pseudojeziku: `ulaz (b); max := b;`. Primijetite da se ove naredbe mogu zamijeniti jednom: `ulaz (max);`

Algoritam rješenja glasi[5]:

```
ulaz (n);
ulaz (b);
max := b;
broj := 1;
za i := 2 do n činiti
     {
          ulaz (b) ;
          ako je b > max onda
               {
                    max := b;
                    broj := 1;
               }
          inače
               ako je b = max onda broj := broj +1;
     }
izlaz (‘Najveći broj bodova je:  ’, max);
izlaz (‘Broj učenika koji su te bodove ostvarili je:  ’, broj);
```

Pretpostavimo da je ispitu pristupilo pet učenika i da su njihovi bodovi redom: 52, 58, 42, 30 i 58. Ove ulazne podatke algoritam će obraditi na sljedeći način:

n	5
b	58, ~~30~~, ~~42~~, ~~58~~, ~~52~~
max	58, ~~52~~
broj	2, ~~1~~, ~~1~~
i	5, ~~4~~, ~~3~~, ~~2~~

Najveći broj bodova je: 58.

Broj učenika koji je te bodove ostvario je: 2.

Podatak da je na ispitu bilo moguće ostvariti najviše 70 bodova nije korišten u algoritmu rješenja. Ponekad znamo ili možemo doznati i više podataka nego što je potrebno da bi riješili problem. Kod stvaranja plana, traženja veze između poznatog i nepoznatog, razlučujemo koji će se podaci upotrijebiti kao ulazni za dani algoritam. Imamo li više podataka nego što nam treba, možemo ih naprosto odbaciti ili upotrijebiti za provjeru ulaznih podataka i/ ili rješenja. Tako podatak da se na ispitu moglo ostvariti najviše 70 bodova, može pomoći da se pri unosu bodova izbjegnu greške.

[5] Istaknute su naredbe kojima je upotpunjen prethodni algoritam koji sada pored najboljeg rezultata ispisuje i broj učenika koji je taj rezultat ostvario.

Zadatak 11.

Napišite program korištenjem pseudojezika za zadani problem.

Obrtnik prodaje suvenire. Cijena suvenira je c kuna, a trošak po jednom suveniru je t kuna. Inicijalni troškovi proizvodnje su p kuna. Kolika je najmanja količina suvenira koju obrtnik treba prodati da bi pokrio svoje troškove?

(Napomena: t je uvijek manji od c. Podaci p, c i t čitaju se s tipkovnice.)

Odgovor:

Budući da je $t < c$ to će svaki prodani suvenir donijeti neku dobit Nakon određenoga broja prodanih suvenira obrtniku će se vratiti uloženi novac i dalja će prodaja za njega predstavljati čisti profit. Primijetimo da se prodajom jednoga suvenira trošak smanji za $c - t$ kuna; dakle je onaj kojega još treba nadoknaditi jednak: $p - (c - t)$ kuna. Sve dok je p pozitivan obrtnik zna da će i sljedeći prodani suvenir služiti za podmirivanje početnoga troška.

Zamislimo stoga fiktivnu prodaju suvenira: prodaju se jedan za drugim dok se ne pokrije trošak. Algoritamskim jezikom ta se prodaja može opisati uzastopnim ponavljanjem naredbe `p:= p-(c-t)` koja se izvodi sve dok je uvjet $p > 0$ istinit. Pobrojimo li koliko je puta naredbu potrebno ponoviti, dobit ćemo odgovor na pitanje: koliko suvenira treba prodati da bi se pokrio trošak.

Algoritam rješenja:

```
ulaz (p,c,t);
k := 0;
dok je p > 0 činiti
{
     p := p - (c-t);
     k := k + 1;
}
izlaz ('Treba prodati ', k , ' suvenira.');
```

Provjera:

Pretpostavimo da je početni trošak $p = 100$ kuna, cijena suvenira c = 34 kune, a trošak po jednom proizvedenom suveniru $t = 10$ kuna. Izlazi da je čista zarada po jednom suveniru 24 kune Prodajom jednoga suvenira početni se trošak smanji na $100 - 24 = 76$ kuna; prodajom dva suvenira smanji se na $76 - 24 = 52$ kune; prodajom tri suvenira jednak je $52 - 24 = 28$ kuna, a prodajom četiri suvenira $28 - 24 = 4$ kune. Dakle, sa pet će suvenira trošak zasigurno biti namiren.

Predloženim će se algoritmom varijable mijenjati na sljedeći način:

p	-20, ~~4~~, ~~28~~, ~~52~~, ~~76~~, ~~100~~
c	34
t	10
k	5, ~~4~~, ~~3~~, ~~2~~, ~~1~~, ~~0~~

Ispisat će se da je broj suvenira koje treba prodati jednak 5, stoga možemo pretpostaviti da je predloženi algoritam ispravan.

Ako najprije matematički "obradimo" problem algoritam rješenja bit će još jednostavniji.

Neka je n broj suvenira koje treba prodati da bi se pokrio trošak. Tada je:

$$p \leqslant n(c-t)$$

Nejednakost podijelimo s $c-t$ i dobijemo:

$$n \geqslant \frac{p}{c-t}$$

Primijetimo da se predznak nije promijenio budući da je $c-t>0$.

Ako je $\frac{p}{c-t}$ cijeli broj, tada je on jednak broju suvenira koji će "točno u kunu" pokriti početni trošak (`n := p div (c-t)`). Ako količnik nije cijeli broj onda ukupni trošak odgovara cjelobrojnom dijelu ovoga količnika (`n :=p div (c-t)`) uvećanom za dio cijene jednoga suvenira. Dakle, broj suvenira koji sigurno pokriva trošak jednak je cjelobrojnom dijelu količnika uvećanom za 1 (`n := p div (c-t)+1`). Algoritam rješenja glasi:

```
ulaz (p,c,t);
n := p div (c-t);
ako je ( p mod (c-t) <> 0 ) onda
     n := n+1;
izlaz ('Treba prodati', n , ' suvenira.');
```

Ovim će se algoritmom za $p=100$, $c=34$ i $t=10$, varijable mijenjati na sljedeći način:

p	100
c	34
t	10
n	5, ~~4~~

Rezultat koji se dobije ovim algoritmom jednak je onome koji smo dobili ranije: obrtnik treba prodati pet suvenira da bi pokrio inicijalni trošak. No ovo je rješenje efikasnije od prethodnog. U prvom algoritmu petlja `dok je` osigurava ponavljanje naredba `p:= p-(c-t); i k:= k+1;`. Broj ponavljanja ovisi o vrijednostima p, c i t. Drugi algoritam rješava problem s dva cjelobrojna dijeljenja, tri zbrajanja i jednim grananjem, neovisno o vrijednostima ulaznih varijabli.

Zadatak 12.

Dinkov tjedni džeparac iznosi t kuna. Početkom tjedna Dinko je u novčaniku imao n kuna. Nakon koliko tjedana može biti siguran da će se taj iznos udvostručiti, uz pretpostavku da neće trošiti više od s kuna tjedno $(s < t)$?

U pseudojeziku napišite program koji će odgovoriti na Dinkovo pitanje. Podaci n, s i t učitavaju se s tipkovnice.

Odgovor:

Neka je u ukupna svota novca s kojom Dinko raspolaže. Njen iznos početkom prvoga tjedna bio je n kuna (`u := n`). Bude li se držao plana Dinko će po isteku svakoga tjedna u novčaniku imati $t-s$ kuna više. Pitanje je nakon koliko će tjedana u novčaniku imati barem dvostruko više novca nego na početku ($u \geqslant 2 \cdot n$)?

Tako dugo dok je ($u < 2 \cdot n$) uštedu od ($t - s$) kuna treba pribrajati ukupnoj svoti (`u:= u + (t-s)`). Budući da svako pribrajanje znači jedan tjedan štednje više to i ukupan broj tjedana b "provedenih u štednji" treba uvećati za 1 (`b:=b+1`).

Algoritam rješenja zapisan u pseudokodu glasi:

```
ulaz (n,s,t);
u := n;
b := 0;
dok je u < 2*n činiti
     {
          u := u + (t-s);
          b := b+1;
     }
izlaz ('Broj tjedana štednje jednak je:  ', b);
```

Pretpostavimo da je Dinko u novčaniku imao iznos od 43 kune. Ako je njegov tjedni džeparac 50 kuna, a tjedno neće trošiti više od 30 kuna, njegova je tjedna ušteda 20 kuna. Nakon koliko će tjedana u novčaniku imati $2 \cdot 43 = 86$ kuna ili više? Nakon prvog će tjedna imati najmanje $43 + 20 = 63$ kune, nakon drugog $63 + 20 = 83$ kune, a nakon tri $83 + 20 = 103$ kune. Dakle, za tri će tjedna u novčaniku imati iznos koji je i veći od dvostrukog ($103 - 2 \cdot 43 = 17$).

Danim algoritmom dobijemo isti rezultat:

n	43
s	30
t	50
u	103, ~~83~~, ~~63~~, ~~43~~
b	3, ~~2~~, ~~1~~, ~~0~~

Algoritamsko rješenje Dinkovog problema bit će jednostavnije ako ga najprije matematički oblikujemo. Uz iste oznake problem se može opisati nejednadžbom:

$$n + b \cdot (t - s) \geqslant 2 \cdot n, \quad s < t; \quad n, b, t, s \in \mathbf{N}$$

u kojoj je b nepoznata veličina. Standardnim postupkom dobijemo: $b \geqslant \dfrac{n}{t - s}$.

Najmanji prirodni broj b koji zadovoljava ovu nejednadžbu jednak je broju tjedana nakon kojih će se svota u Dinkovu novčaniku sigurno udvostručiti. Iz ovih razmatranja proizlazi sljedeći algoritam:

```
ulaz (n,s,t);
b := n div (t-s);
ako je (n mod (t-s)) <> 0 onda b := b+1;
izlaz ('Broj tjedana štednje jednak je:  ', b);
```

Zadatak 13.

Napišite program u pseudojeziku koji će za učitani cijeli broj n ispisati koliko taj broj ima nula. Na primjer, za broj 50803 izlaz će biti 2, a za 517, bit će 0.
Broj n učitava se s tipkovnice.

Odgovor:

Uspijemo li izdvojiti pojedine znamenke broja n, bit će jednostavno prebrojiti koliko među njima ima nula. Budući da je broj zapisan u dekadskom brojevnom sustavu, rješenje ćemo potražiti u cjelobrojnom dijeljenju toga broja s 10. Primjerice, za $n = 517$ dobijemo:

```
517 div 10 = 51
517 mod 10 = 7
```

Vidimo da cjelobrojno dijeljenje s 10 skraćuje broj za krajnju desnu znamenku, a da je ostatak toga dijeljenja jednak krajnjoj desnoj znamenci.

Problem ćemo riješiti tako da najprije izdvojimo i ispitamo krajnju desnu znamenku broja (`n mod 10`), a potom ju "izbrišemo" (`n := n div 10`). Ovaj postupak treba ponavljati sve dok broju n ne "izbrišemo" sve znamenke.

Algoritam rješenja:

```
ulaz (n);
broj := 0;
ponavljati
      ako je (n mod 10) = 0 onda broj := broj +1;
      n := n div 10;
do n = 0;
izlaz ('Broj nula je jednak:  ', broj);
```

Ovim će se algoritmom za $n = 50803$, varijable mijenjati na sljedeći način:

n	0, ~~5~~, ~~50~~, ~~508~~, ~~5080~~, ~~50803~~
broj	2, ~~1~~, ~~0~~

Broj nula je jednak: 2.

Zadatak 14.

Tijekom obrade, binarno kodirani podaci prenose se između jedinica računalnog sustava. Ponekad se dogodi greška; zamijeni se jedan ili više bitova u kodnoj riječi. Za otkrivanje takve vrste grešaka koristi se **paritetni bit** koji se dodaje kodiranom podatku tako da ukupan broj jedinica bude paran[6] Primjerice, ako je podatak predstavljen binarnim kodom 0001010, onda će uz dodatak paritetne zaštite biti: **0**0001010, dok će podatak 0001011 uz dodatak iste zaštite biti: **1**0001011.

Na ovaj se način postiže da svaki podatak ima paran broj jedinica. Ako sustav tijekom rada "otkrije" podatak s neparnim brojem jedinica tada "zna" da taj podatak nije kodiran kako valja. Ako paritetna provjera i ne otkrije grešku to ne znači da je nema. Naime, ukoliko je promijenjen samo jedan bit, paritetnom će se provjerom takva greška otkriti, ali ukoliko su dva bita promijenjena ova provjera neće dati rezultata.

U pseudojeziku napišite program koji omogućuje provjeru pariteta. Odgovarajući binarni podatak b čita se s tipkovnice. Broj jedinica učitanog broja mora biti paran. Ukoliko nije, treba ispisati poruku da je otkrivena greška.

[6] Kažemo da je paritet *paran*. Moguće je zahtijevati da broj jedinica bude neparan. Tada govorimo o *neparnom* paritetu.

Odgovor:

Broj jedinica binarnoga broja b jednak je zbroju svih znamenaka toga broja. Stoga izdvajamo jednu po jednu znamenku broja b i pribrajamo je varijabli `zbroj`. Postupak:

— varijabli `zbroj` pribroji se krajnja desna znamenku broja b;

— broj b "skrati se" za krajnju desnu znamenku;

ponavljamo sve dok se broj b ne izjednači s nulom. Preostaje vidjeti je li u varijabli `zbroj` pohranjen paran ili neparan broj:

— ako je neparan, onda je paritetnom provjerom otkrivena greška;

— ako je paran, paritetna provjera nije otkrila grešku.

Algoritam rješenja:

```
ulaz (b);
zbroj := 0;
ponavljati
     zbroj := zbroj + b mod 10;
     b := b div 10;
do b = 0;
ako je (zbroj mod 2) = 0 onda
     izlaz ('Greška nije otkrivena.')
inače
     izlaz ('Otkrivena je greška.');
```

Zadatak 15.

Borna ima n bojica. Slaže ih u hrpice tako da u svakoj bude jednak broj bojica. Odlučio je da u svakoj hrpici budu barem dvije bojice. Na koliko načina to može učiniti? Napišite program korištenjem pseudojezika za zadani problem. (Napomena: broj bojica n čita se s tipkovnice.)

Odgovor:

U svakoj će hrpici biti jednak broj bojica jedino ako je broj n djeljiv s brojem hrpica h. Svaka hrpica tada ima točno `n div h` bojica. Dakle treba pobrojati sve brojeve h koji dijele n tako da `(n div h)` $\geqslant 2$.

Uvjet `(n div h)` $\geqslant 2$ je matematički zapis tvrdnje "hrpica ima barem dvije bojice". No ovu smo tvrdnju mogli izreći i drukčije: "hrpica ne smije imati samo jednu bojicu" što je opet isto kao da smo rekli "broj hrpica ne smije biti jednak broju bojica". Proizlazi da uvjet `(n div h)` $\geqslant 2$ možemo zapisati kraće: $h \neq n$.

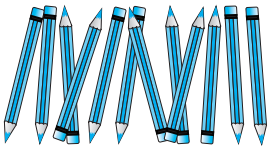
Preostaje među brojevima od 1 do $n-1$ pronaći i pobrojati djelitelje broja n. Budući da u intervalu `n div 2` do `n-1` nema djelitelja broja n, djelitelje ćemo tražiti među brojevima od 1 do `n div 2` i tako skratiti postupak pretraživanja.

Algoritam rješenja:

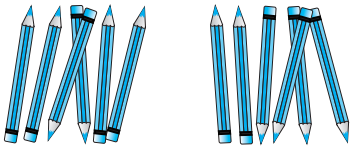
```
ulaz (n);
broj := 0
za i := 1 do n div 2 činiti
     ako je (n mod i) = 0 onda broj := broj + 1;
izlaz ('Borna bojice može složiti na ',broj,' načina.');
```

Provjera:

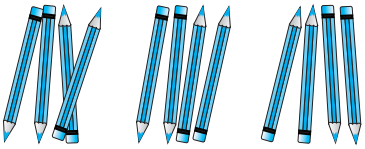
12 bojica Brona može složiti na pet različitih načina:



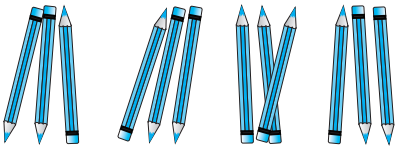
jedna hrpica - 12 bojica



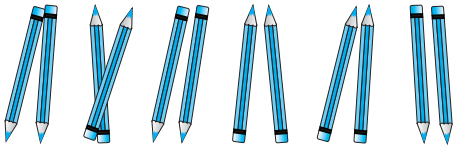
dvije hrpice - 6 bojica svaka



tri hrpice - 4 bojice svaka



četiri hrpice - 3 bojice svaka



šest hrpica - 2 bojice svaka

Do istog će zaključka doći i predloženi algoritam.

Zadatak 16.

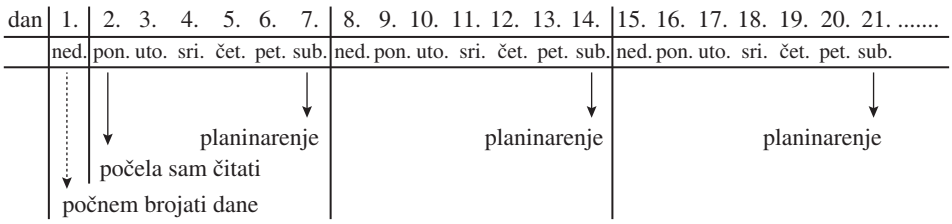
Knjiga ima *n* stranica. Svakoga dana planiram pročitati 20 stranica knjige, osim subote koja je predviđena za planinarenje. Počnem li čitati u ponedjeljak, nakon koliko ću dana pročitati cijelu knjigu?

U pseudojeziku napišite program koji će dati odgovor na ovo pitanje. Broj stranica knjige *n* čita se s tipkovnice.

Odgovor:

Algoritam je jednostavan: svakoga dana od ukupnog broja stranica knjige oduzme se 20 stranica (`n := n-20`). Naravno, subota se preskače. Ovaj se postupak ponavlja tako dugo dok još ima nepročitanih stranica knjige (do $n \leqslant 0$). No kako odrediti koji je dan subota?

Neka je `d` varijabla koja "broji" dane počevši od dana kad sam počela čitati knjigu. Taj dan je ponedjeljak. Knjigu ću čitati pet dana (ponedjeljak, utorak, srijeda, četvrtak, petak), a šesti ću dan preskočiti. Dalje ću čitati u pravilnim intervalima: šest dana čitam (nedjelja, ponedjeljak, utorak, srijeda, četvrtak, petak), a subotu, sedmi dan po redu preskačem. Počnem li dane brojati od nedjelje (v. sliku), interval čitanja uvijek je isti: šest dana čitam, sedmi planinarim. Algoritam je sad jednostavniji, ali se broj dana povećao za 1. Stoga prije ispisa rezultata to treba "popraviti" naredbom: `d := d-1`.



Algoritam rješenja:

```
ulaz (n);
d := 1;
ponavljati
      d := d + 1;
      ako je (d mod 7) <>0 onda n := n-20;
do n <= 0;
d := d - 1;
izlaz (' Knjigu ću pročitati za ', d , ' dana.');
```

Provjera:

Pretpostavimo da knjiga ima 150 stranica. Od ponedjeljka do petka pročitat ću 100 stranica. Ostaje mi još 50 stranica koje ću po danima rasporediti na sljedeći način: 20 stranica u nedjelju i ponedjeljak te 10 u utorak. Znači ukupno mi treba 9 dana.

Pratimo li na koji se način mijenjaju varijable, vidimo da isti rezultat dobijemo i predloženim algoritmom

`n`	$-10, \not{10}, \not{30}, \not{50}, \not{70}, \not{90}, \not{110}, \not{130}, \not{150}$
`d`	$9, \not{10}, \not{9}, \not{8}, \not{7}, \not{6}, \not{5}, \not{4}, \not{3}, \not{2}, \not{1}$

Zadatak 17.

Dva autobusa zajedno kreću s iste polazne stanice. Ako prvome treba n, a drugome m minuta da se vrati na polazište, nakon kojeg će se vremena ponovo naći na polaznoj stanici?
U pseudojeziku napišite program koji će ispisati traženo vrijeme u satima i minutama. Ulazni podaci n i m čitaju se s tipkovnice.

Odgovor:

Nakon p minuta prvi se autobus od polazne stanice udaljio za `p mod n` minuta vožnje, a drugi za `p mod m` minuta. Da bi se prvi autobus vratio u polaznu stanicu treba biti `(p mod n) = 0`. Slično, drugi će se autobus vratiti u polaznu stanicu kad je `(p mod m) = 0`. Zanima nas kad će se oba autobusa naći u polazištu, to jest, za koji će p biti i `(p mod n)=0`, i `(p mod m) =0`. Naravno, trivijalni slučaj da je $p = 0$ isključujemo.

Odaberemo li p tako da bude jednak umnošku vrijednosti n i m, postavljeni će uvjeti sigurno biti ispunjeni. No postoji li manji p koji ispunjava dane uvjete? Drugim riječima hoće li se "susret" autobusa dogoditi za manje od $n \cdot m$ minuta?

To ćemo utvrditi uzastopnom provjerom tako da vrijednost p postepeno povećavamo za 1 i taj postupak ponavljamo tako dugo dok oba uvjeta ne budu ispunjena. Pretpostavljamo da su autobusi sigurno napravili barem jedan "puni krug vožnje", pa je p sigurno veći ili jednak i od n i od m. Stoga provjeru možemo početi tako da za p odaberemo jedan od tih brojeva.

Algoritam rješenja:

```
ulaz (n,m);
p := n-1;
ponavljati
        p := p+1;
do ( (p mod n) = 0 ) I ((p mod m)= 0);
izlaz ('Autobusi će se na polaznoj stanici naći za ',
        p div 60 , ' sati i ', p mod 60 , ' minuta.');
```

Provjera rješenja:

Da bismo lakše pratili na koji način predloženi algoritam mijenja varijable, za ulaz odaberemo "male" brojeve, iako oni nisu realni. Primjerice, $n = 6$ i $m = 4$. Algoritam bi trebao ispisati da će se autobusi za 12 minuta ponovo naći u polaznoj stanici.

Pratimo li promjenu varijabli vidimo da će algoritam za navedene vrijednosti dati ispravni rezultat:

n	6
m	4
p	12, ~~11~~, ~~10~~, ~~9~~, ~~8~~, ~~7~~, ~~6~~, ~~5~~

Autobusi će se na polaznoj stanici naći za 0 sati i 12 minuta.

Koristeći matematičku terminologiju, možemo reći da će se autobusi zajedno naći u polaznoj stanici nakon p minuta, za svaki p koji je zajednički višekratnik brojeva n i m. Budući da nas zanima kada će se to prvi put dogoditi, treba naći *najmanji zajednički višekratnik* brojeva n i m. Uobičajeni je postupak da pomnožimo proste faktore i jednog i drugog broja pri čemu faktore koji se ponavljaju uzmemo onoliko puta koliko se najviše nalaze u jednome od brojeva. Kada smo računali "ručno" taj smo postupak zapisivali na sljedeći način (primjerice za $n = 8$, $m = 12$):

8	12	2
4	6	2
2	3	2
1	3	3

S desne strane crte zapišemo jedan od faktora brojeva n ili m. Ako je broj djeljiv s tim faktorom, podijelimo ga, ako ne, tada ga prepišemo i nastavljamo postupak u sljedećem retku. Kada smo na taj način izdvojili sve proste faktore postupak je gotov. Najmanji zajednički višekratnik brojeva n i m jednak je umnošku izdvojenih faktora. U ovom je primjeru: $p = V(8, 12) = 2 \cdot 2 \cdot 2 \cdot 3 = 24$.

Zapišemo li ovaj postupak u pseudojeziku, dolazimo do još jednog rješenja danog problema.

```
ulaz (n,m);
p := 1;
k := 2;
dok je  (n>1) ILI (m>1) činiti
{
     dok je ((n mod k)= 0) ILI ((m mod k) = 0) činiti
     {
          p := p*k;
          ako je (n mod k) = 0 onda n := n div k;
          ako je (m mod k) = 0 onda m := m div k;
     }
```

```
        k := k+1;
    }
    izlaz ('Autobusi će se na polaznoj stanici naći za ',
              p div 60 , ' sati i ', p mod 60 , ' minuta.');
```

Provjerom za ulazne vrijednosti $n = 6$ i $m = 4$ vidimo da algoritam daje isti izlaz kao i prethodni.

n	1, ~~3~~, ~~6~~
m	1, ~~2~~, ~~4~~
k	4, ~~3~~, ~~2~~
p	12, ~~4~~, ~~2~~, ~~1~~

Zadatak 18.

Na koliko se različitih načina iznos od k kuna može isplatiti u kovanicama od 2 i 5 kuna? Napišite program korištenjem pseudojezika za zadani problem. Podatak k čita se s tipkovnice.

Odgovor:

Da bi izvršili isplatu s n novčanica od 2 kune i s m novčanica od 5 kuna, treba biti

$$2 \cdot n + 5 \cdot m = k$$

Pritom broj kovanica od dvije kune sigurno nije veći od `k div 2`. Također, broj kovanica od pet kuna nije veći od `k div 5`. Znači, treba pronaći takve cijele brojeve n i m da vrijedi:

$$2 \cdot n + 5 \cdot m = k$$

$$0 \leqslant n \leqslant k \text{ div } 2$$

$$0 \leqslant m \leqslant k \text{ div } 5$$

Zadatak će biti riješen kad prebrojimo sve moguće kombinacije brojeva n i m koje zadovoljavaju dane uvjete. Neka je b brojač takvih kombinacija. Početna vrijednost brojača je 0. Svaki put kad pronađemo odgovarajuću kombinaciju ta se vrijednost poveća za 1. Može se dogoditi da ne pronađemo niti jednu kombinaciju koja zadovoljava dane uvjete. U tom se slučaju iznos od k kuna ne može isplatiti u kovanicama od 2 i 5 kuna.

Rješenje možemo dobiti sljedećim algoritmom.

```
ulaz (k);
b := 0;
za n := 0 do k div 2 činiti
     za m := 0 do k div 5 činiti
          ako je (2*n+5*m=k) onda b := b+1;
ako je b = 0 onda
     izlaz ('Isplata nije moguća.  ')
inače
     izlaz ('Isplata se može izvršiti na ', b, ' načina.')
```

Primjerice, iznos od 12 kuna možemo isplatiti na 2 načina: sa 6 kovanica od dvije kune ili sa dvije kovanice od 5 kuna i jednom kovanicom od 2 kune.

Predloženim algoritmom dobijemo isto rješenje:

k	12
b	2, 1, 0
n	6, 5, 4, 3, 2, 1, 0
m	2, 1, 0, 2, 1, 0, 2, 1, 0, 2, 1, 0, 2, 1, 0, 2, 1, 0, 2, 1, 0

Ugniježđena petlja za u danom je algoritmu poslužila da se sistematično "popišu" sve moguće kombinacije vrijednosti varijabla n i m kako bi se za svaku od njih provjerila jednakost $2 \cdot n + 5 \cdot m = k$. Broj mogućih kombinacija, dakle broj provjera jednak je:

$$((k \text{ div } 2) + 1) \cdot ((k \text{ div } 5) + 1)$$

No taj se broj može smanjiti. Pretpostavimo da je poznato koliki će udio kovanica od dvije kune biti u isplati. Neka je taj broj n. Onda kovanicama od 5 kuna preostaje podmiriti $(k - 2 \cdot n)$ kuna. Dakle je $0 \leqslant m \leqslant (k - 2 \cdot n)$ div 5.

Ugradimo li ovo saznanje u postojeći algoritam, dobijemo:

```
ulaz (k);
b := 0;
za  n := 0 do k div 2 činiti
      za m := 0 do (k-2*n) div 5 činiti
            ako je 2*n+5*m=k onda b := b+1;
ako je b= 0 onda
      izlaz (‘Isplata nije moguća.’)
inače
      izlaz (‘Isplata se može izvršiti na ’, b, ‘ načina.’)
```

Ovaj je algoritam efikasniji od prethodnog jer je manji broj kombinacija varijabla n i m koje se uzimaju u razmatranje. Za $k = 12$ ovim se algoritmom varijable mijenjaju na sljedeći način:

k	12
b	2, 1, 0
n	6, 5, 4, 3, 2, 1, 0
m	0, 0, 0, 1, 0, 1, 0, 2, 1, 0, 2, 1, 0