UNIVERZITET CRNE GORE
EKONOMSKI FAKULTET

ALGORITMI

- Dopunjeno izdanje -

Autori:
Prof. dr Ljiljana Kašćelan
Doc. dr Biljana Rondović
Mr Tamara Đuričković

Podgorica, 2010. godine

SADRŽAJ

ZADATAK I ALGORITAM

Postoji veliki broj zadataka koje čovjek treba da riješi pomoću računara. Pod procesom rješavanja zadatka na računaru podrazumijeva se zajednička djelatnost čovjeka i računara. Značajna sposobnost ljudi jeste da uoče zadatak, da ga dobro postave, a zatim da ga riješe. Uočiti zadatak znači ocijeniti, zaključiti da na osnovu nekih poznatih veličina, ima smisla tražiti, odrediti, neke druge nepoznate veličine. Ove poznate veličine zovemo *polazne* veličine zadatka, a tražene, nepoznate veličine, zovemo *rješenja* zadatka. Kada su uočene polazne veličine i svrha zadatka, formuliše se *postavka* zadatka, u kojoj se ovo jasno navodi.

Uslovno se može proces rješavanja zadatka na računaru predstaviti kroz nekoliko etapa:

1. Formulacija problema
2. Matematički oblik problema
3. Algoritmizacija problema
4. Programiranje
5. Izrada test primjera
6. Testiranje problema
7. Dobijanje i analiza rezultata

Ne zanemarujući značaj nijedne od navedenih etapa na vježbama će posebna pažnja biti posvećena problemu algoritma i načinima njihovog rješavanja.

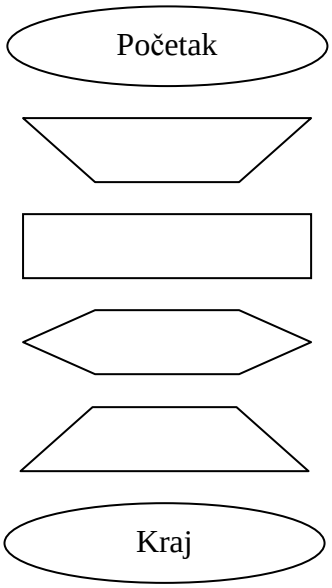
ALGORITAM

Def: *Algoritam je skup pravila formulisanih za rješavanje nekog zadatka.*

Najčešće je algoritam predstavljen u obliku blok šeme sa jasno definisanim nizom radnji. Grafički zapis algoritma naziva se algoritamska šema. Algoritamska šema se odlikuje sljedećim karakteristikama:

- Omogućuje zapis algoritma na način koji obezbjeđuje lako otkrivanje grešaka u strukturi algoritma
- Omogućuje kraći i jasniji zapis algoritma (prednost u odnosu na tekstualni oblik)
- Pregledna je veza između detalja i cjeline algoritma
- Algoritam u ovom obliku je nezavisan od njegovog daljeg korišćenja.

Grafički simboli koji se koriste za algoritamske šeme mogu se prikazati na sledeći način:



Definiše početak (prvi algoritamski korak)

Definiše ulazne veličine algoritma

Definiše obradu podataka

Uslovni algoritamski korak

Definiše izlazne veličine algoritma

Definiše kraj algoritma

Primjer la:

Poznavajući osobine prirodnih brojeva i operacije sabiranja i množenja prirodnih brojeva, riješiti sljedeći zadatak (tekstualno i grafički)

Naći proizvod prirodnih brojeva X i Y.

Z=X*Y

Ako je X=2 i Y=5 $\Rightarrow \underbrace{2+2+2+2+2}_{5 \text{ puta}}$ $\quad X*Y=\underbrace{X+X+X+X+X..}_{Y \text{ puta}}$

,

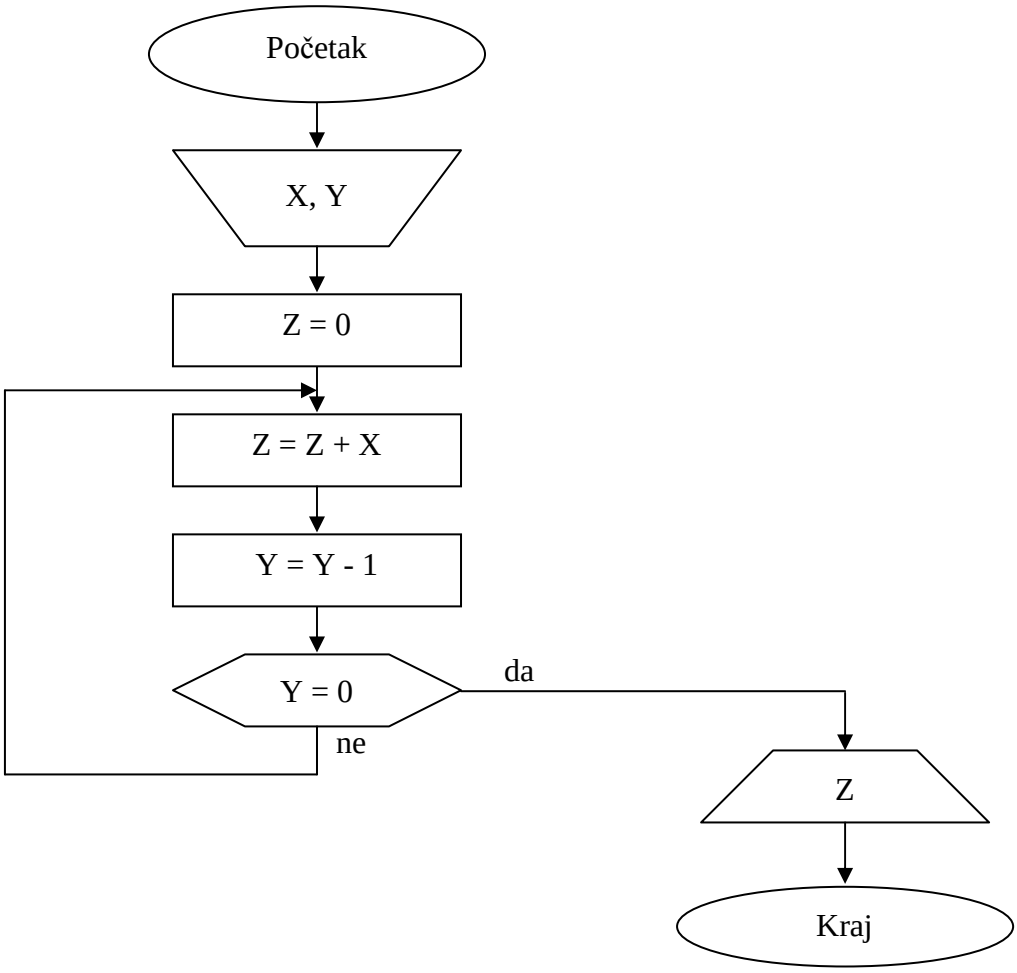
Koraci za rješavanje bi bili :

1. Ulazne veličine su X i Y, pređi na korak 2
2. Postaviti da je Z=O, pređi na korak 3
3. Uvećaj Z za X, pređi na korak 4
4. Umanji Y za 1, pređi na korak 5
5. Ako je Y≠0 (1,2,3…) vrati se na korak 3
6. Ako je Y=0 pređi na korak 6
7. Izlazna veličina je Z. KRAJ.

Kako to izgleda u konkretnom slučaju?

1. X = 15, Y =3
2. Z = 0
3. Z = Z + X$\Rightarrow$ 0 + 15 = 15
4. Y = Y – 1 $\Rightarrow$ 3 – 1 = 2
5. Y $\neq$ 0
3. Z = Z + X $\Rightarrow$ 15 + 15 = 30
4. Y = Y – 1 $\Rightarrow$ 2 - 1 = 1
5. Y $\neq$ 0
3. Z = Z + X $\Rightarrow$ 30 + 15 = 45
4. Y = Y – 1 $\Rightarrow$ 1 – 1 = 0
5. Y = 0
6. Z = 45 KRAJ

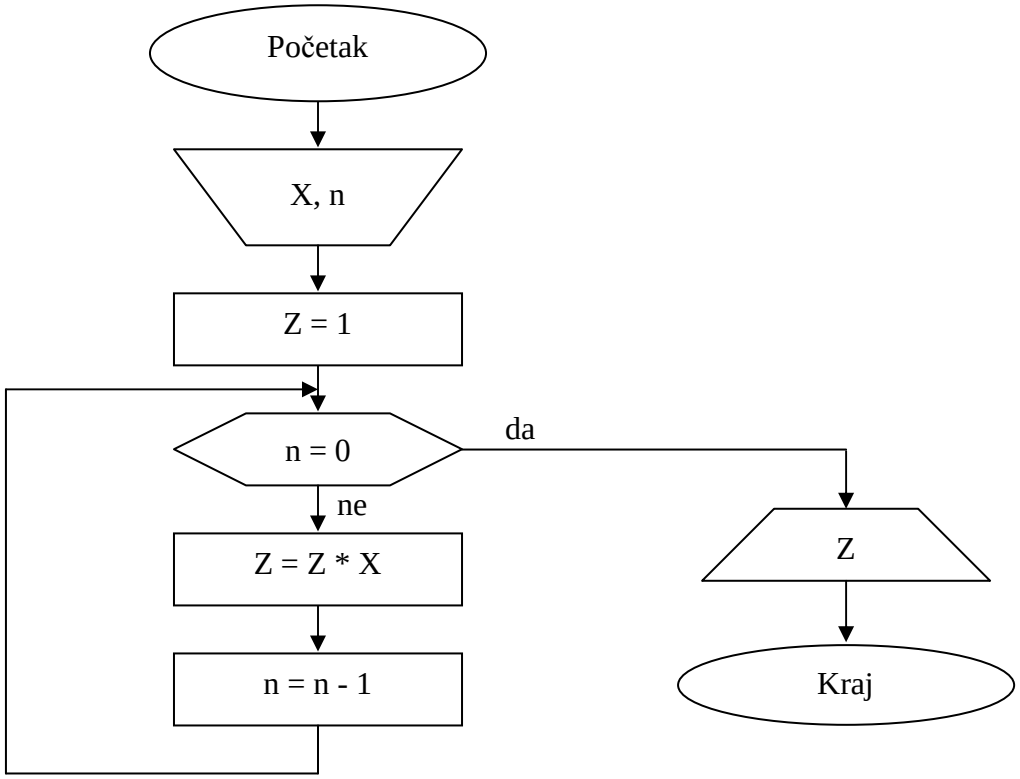
Grafički zapis algoritma:



Primjer 1b:

Sastaviti algoritamsku šemu za stepenovanje prirodnog broja X prirodnim brojem N (X^n), koristeći operaciju množenja

$Z = X^n \Rightarrow Z = X * X * X * X * X * X * \ldots X$ (n puta)



VRSTE ALGORITAMSKIH ŠEMA

Algoritamske šeme se mogu podijeliti u tri kategorije:

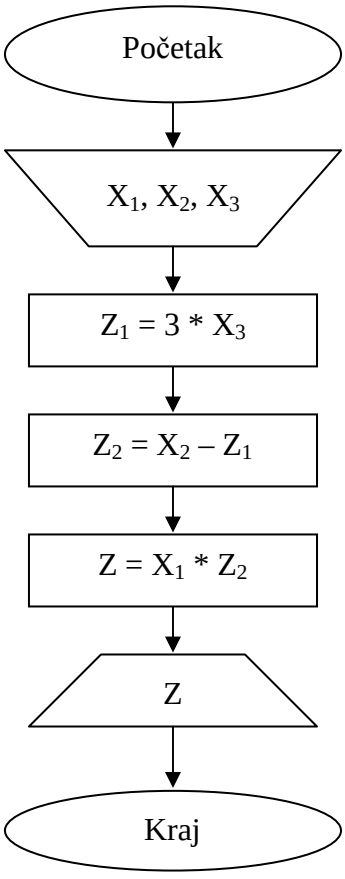
- Linijske algoritamske šeme
- Ciklične algoritamske šeme
- Složene algoritamske šeme

1) ***Linijske algoritamske šeme*** su one šeme kod kojih se svaki algoritamski korak izvršava najviše jedanput u toku izvršavanja algoritma. Mogu biti proste i razgranate.
   a. Proste linijske algoritamske šeme su šeme kod kojih se svaki algoritamski korak izvršava tačno jednom u toku jednog izvršavanja algoritma. Sastoji se od algoritamskih koraka ulaza, obrade i izlaza.

Primjer 1c.

Sastaviti algoritamsku šemu za izračunavanje vrijednosti Z po sljedećoj formuli: $Z = X_1 * (X_2 - 3X_3)$, tako da u jednom algoritamskom koraku može biti samo jedna aritmetička operacija.

Rješenje:

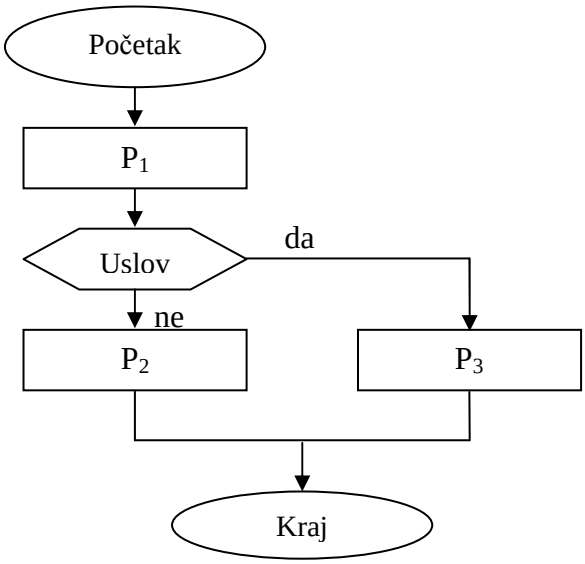


Ulazne veličine su X_1, X_2, X_3
Međurezultati su Z_1, Z_2, Z
Izlazna veličina je Z

b. Razgranate linijske šeme. Za razliku od prostih linijskih šema kod kojih se svaki algoritamski korak izvršava tačno jednom, kod razgranatih linijskih šema svaki algoritamski korak se izvršava najviše jednom (znači jednom ili nijednom) i obavezno sadrži bar jedan uslovni algoritamski korak (vidjeli smo u prethodnom primjeru da nije bilo uslovnih algoritamskih koraka). Ako je uslov ispunjen, izlaz iz algoritamskog koraka biće označen sa 'da', a ako uslov nije ispunjen izlaz će biti označen sa 'ne', ili će biti bez oznake.

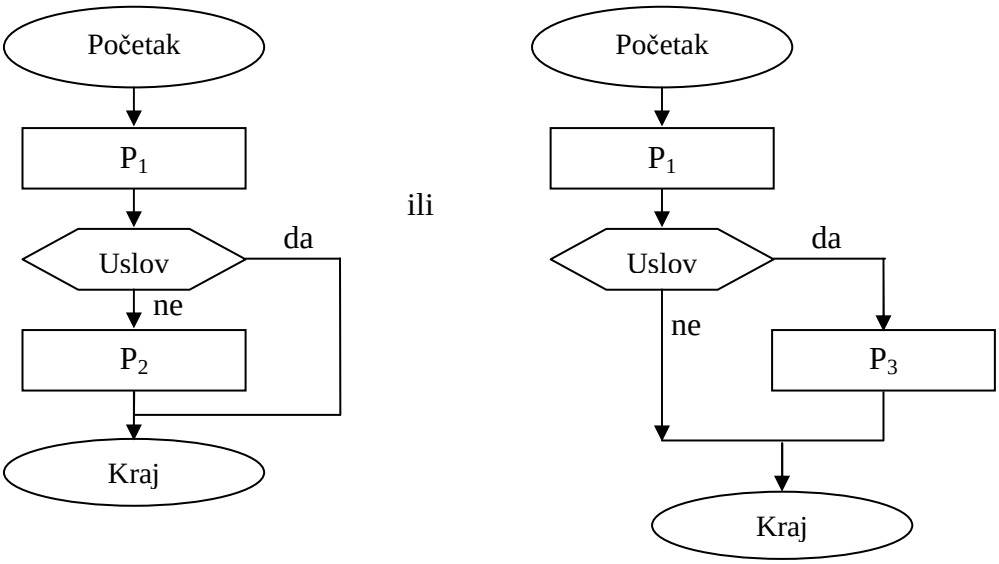
Može se reći da su razgranate linijske šeme sastavljene od 3 proste šeme i uslovnog algoritamskog koraka.

Kako to izgleda?



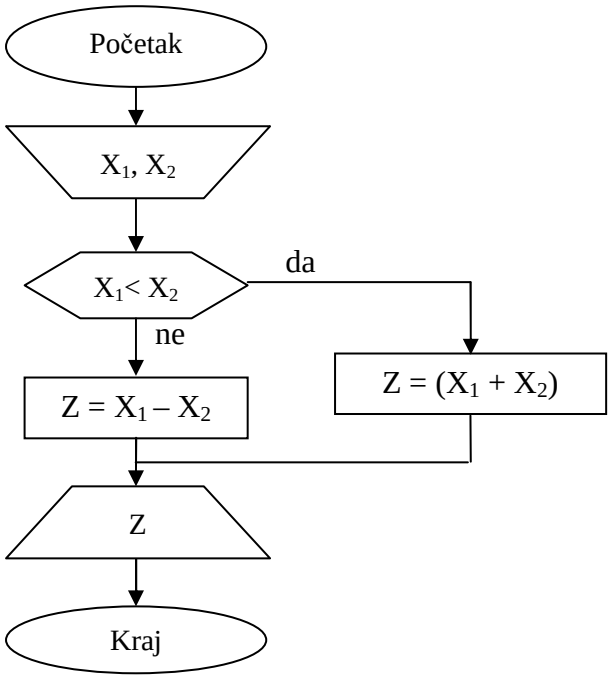
U toku jednog izvšavanja algoritma izvršiće se samo jedna od prostih šema (P_2 ili P_3). Neka od šema P_2 ili P_3 može biti izostavljena.

Znači, algoritam može imati sledeći oblik:



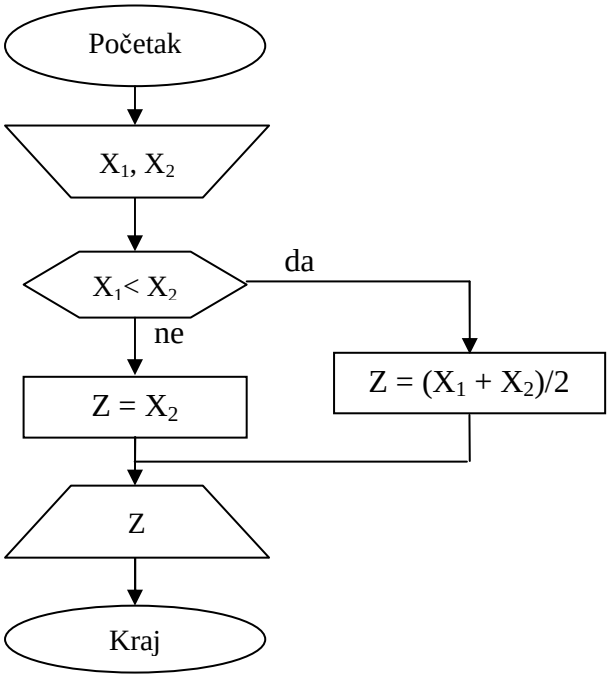
Primjer 1d, za razgranate linijske šeme:
Sastaviti algoritam za izračunavanje vrednosti Z po formuli:

$$Z = \begin{cases} X_1 + X_2;\ X_1 < X_2 \\ X_1 - X_2;\ X_1 \geq X_2 \end{cases}$$



Primjer 1e, za razgranate linijske šeme.
Sastaviti algoritam za izračunavanje vrijednosti Z. Ako je X_1 ocjena iz marketinga, X_2 ocjena iz informatike, izračunati Z po formuli:

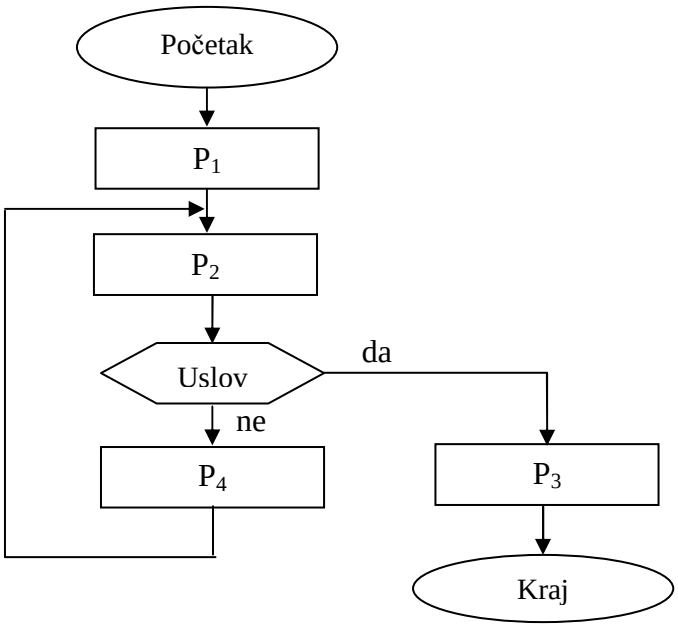
$$Z = \begin{cases} \text{Prosječna ocjena},\ X_1 < X_2 \\ \text{Ocjena iz Informatike};\ X_1 \geq X_2 \end{cases}$$



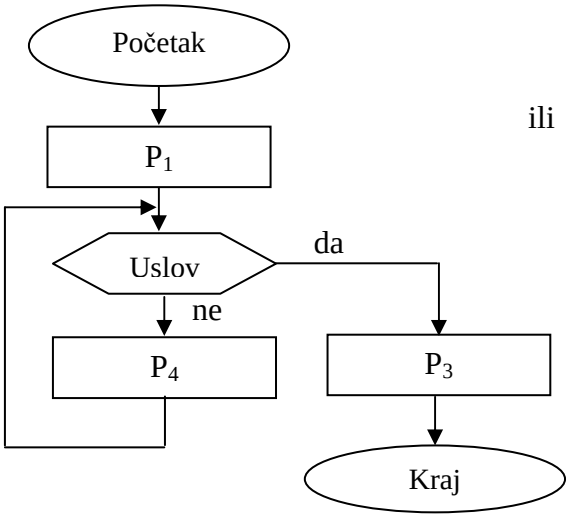
2) ***Ciklične algoritamske šeme*** su one šeme kod kojih se jedan ili više algoritamskih koraka može izvršavati više od jedanput u toku jednog izvršavanja algoritma.

Ovi algoritamski koraci čine ciklus. Uopšte rečeno, ciklične šeme se sastoje od prostih šema P_1, P_2, P_3, P_4 i uslovnog algoritamskog koraka. Ukoliko je uslov ispunjen, vrši se izlazak iz ciklusa. U suprotnom, ciklus se ponavlja. Dakle, uslov definiše izlazak iz ciklusa i zove se izlazni kriterijum ciklusa. Obično je to broj ponavljanja ciklusa.

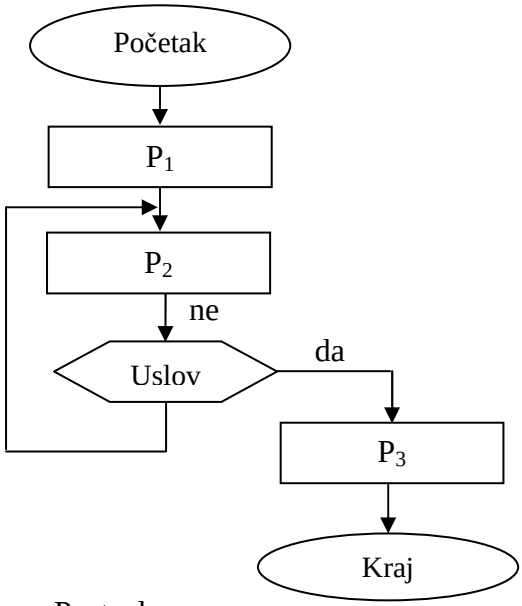
Kako to izgleda?



Neka od prostih šema P_2 ili P_4 može biti izostavljena.



ili



Preuslov
P_4 se može izvršiti 0 ili više puta

Postuslov
Koraci unutar ciklusa (P_2) će se izvršiti najmanje jedanput

Ciklične algoritamske šeme mogu biti konstante i promjenljive.

a) Konstantne ciklične šeme su šeme kod kojih se zakon obrade unutar ciklusa ne mijenja.

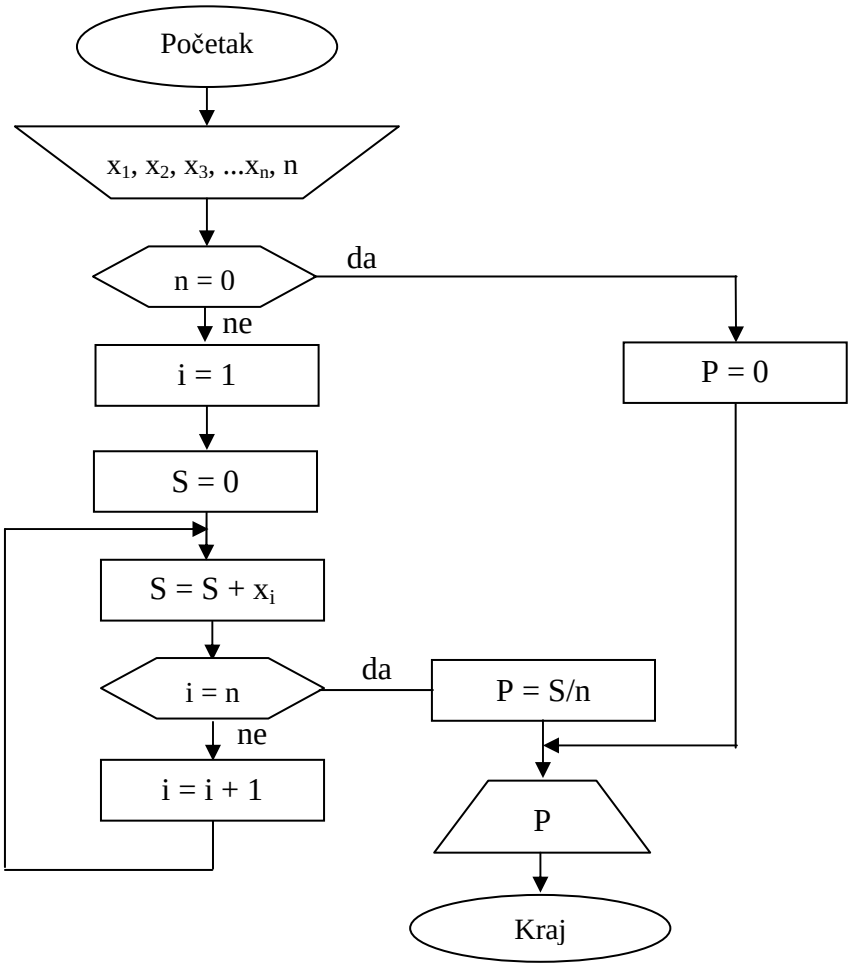
Primjer 2a, za konstantne ciklične šeme

Sastaviti algoritam koji za poznato n i brojeve x_1, x_2, x_3, … x_n računa prosječnu vrijednost brojeva.

Rješenje:

$P = (x_1 + x_2 + x_3 + \ldots + x_n)/n$

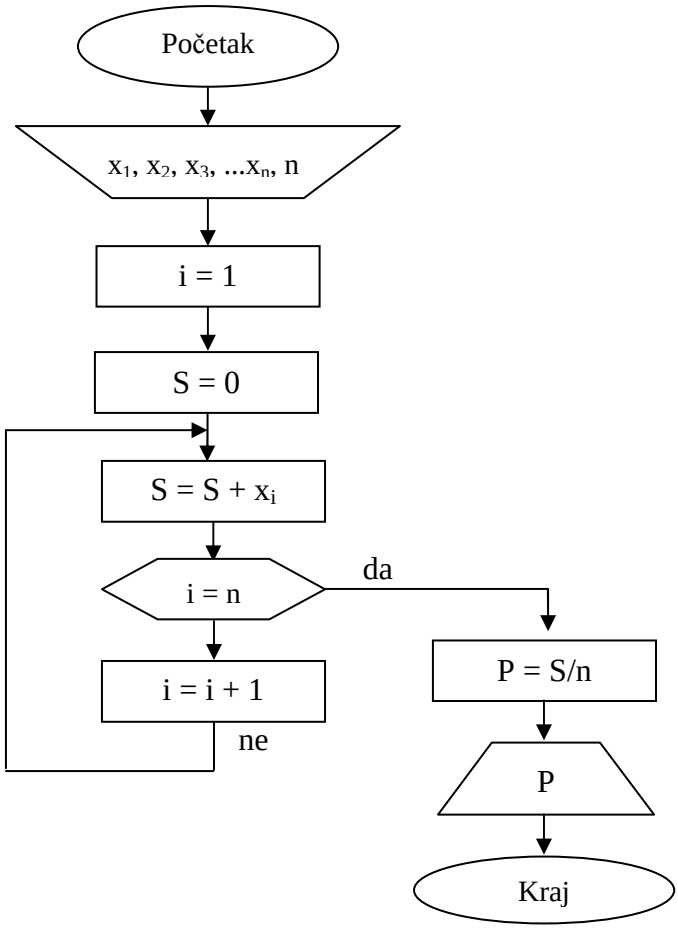
$P = S/n \quad (S = x_1 + x_2 + x_3 + \ldots + x_n)$



Primjer 2a*

Da je zadatak glasio:

Sastaviti algoritam koji za poznato n>0 i brojeve x_1, x_2, x_3, … x_n računa prosječnu vrijednost brojeva, onda bi algoritamska šema izgledala:



b) Promjenljive ciklične šeme su šeme kod kojih u ciklusu dolazi do promjene zakona obrade u toku izvršavanja algoritma.

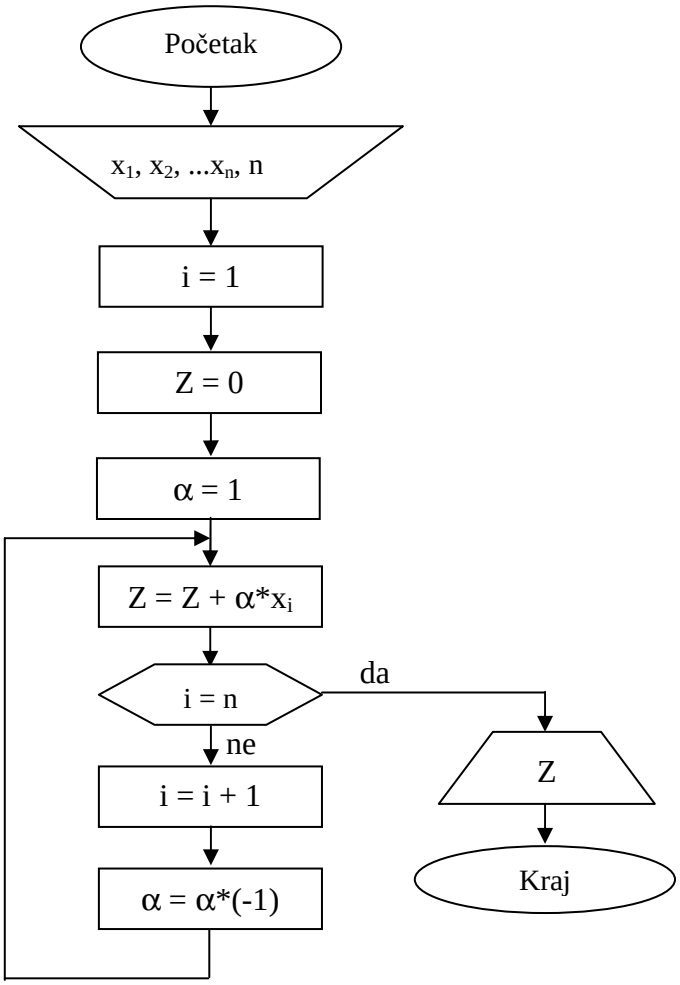
Primjer 2b, za promenljive ciklične šeme:

Sastaviti algoritam koji izračunava sumu:

$$Z = \sum_{i=1}^{n} (-1)^{i+1} * x_i$$ za poznato n>o i x_1, x_2, …x_n

Kada je stepen neparan $\Rightarrow (-1)^3 = (-1)*(-1)*(-1) = -$ (minus)
Kada je stepen paran $\Rightarrow (-1)^2 = (-1)*(-1) = +$ (plus)

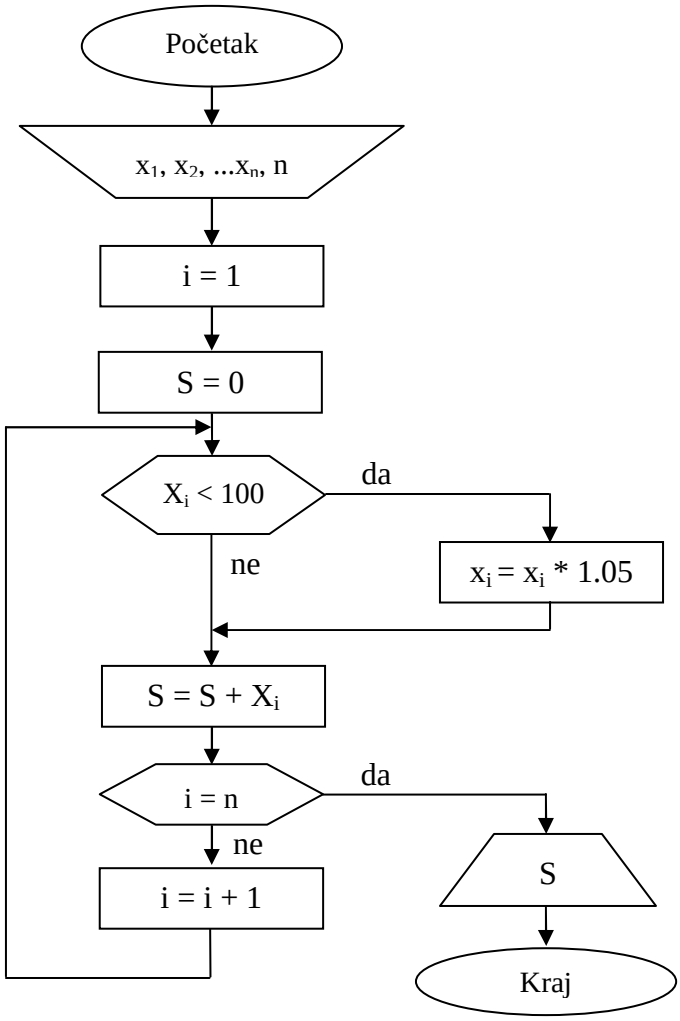
U zadatku ćemo izraz $(-1)^{i+1}$ obilježiti sa α
$Z = X_1 - X_2 + X_3 - X_4 + X_5 - X_6$ …


Početak
x1, x2, ...xn, n
i = 1
Z = 0
α = 1
Z = Z + α*xi
i = n
da
ne
i = i + 1
α = α*(-1)
Z
Kraj

3) ***Složene algoritamske šeme*** su šeme koje se dobijaju kompozicijom gore navedenih šema.

Primjer 3a, za složene algoritamske šeme

Sastaviti algoritam koji za poznato n>0 i brojeve $x_1, x_2, x_3, \ldots x_n$ računa sumu brojeva $x_1, x_2, x_3, \ldots x_n$ uvećanih za 5% u slučaju da su manji od 100.



EKVIVALENTNI ALGORITMI I SLOŽENOST ALGORITMA

Za rješavanje jednog zadatka može se sastaviti više različitih algoritama, a da pri tom svaki od njih bude tačan. Za takve algoritme kažemo da su ekvivalentni. Ekvivalentni algoritmi za iste ulazne veličine daju iste izlazne veličine. Među ekvivalentnim algoritmima treba izabrati onaj koji najefikasnije rješava zadatak. Kriterijumi za efikasnost algoritma koji se rješava na računaru su minimalan utrošak memorije i velika brzina izvršavanja, složenost strukture algoritama itd.

Utrošak memorije (tj. broj koji pokazuje broj promenljivih koje koristi algoritam) naziva se prostorna složenost algoritma.

Primjer:

Algoritamska šema za zadatak koji smo radili i koji je glasio:

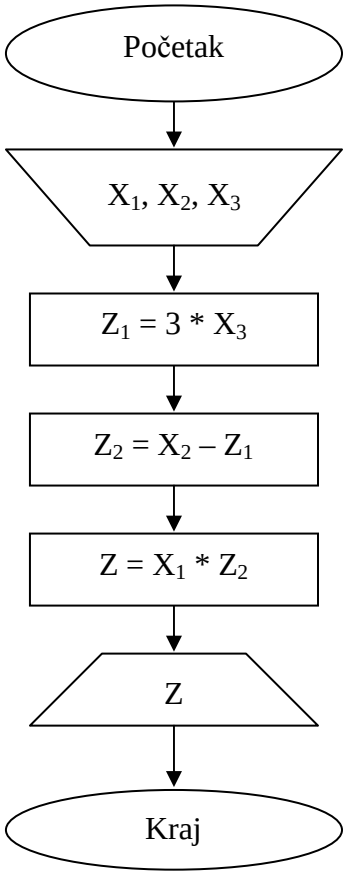
Sastaviti algoritamsku šemu za izračunavanje vrijednosti Z po formuli:

$Z = X_1*(X_2-3X_3)$, tako da u jednom algoritamskom koraku može biti samo jedna aritmetička operacija, je tačna ali nije optimalna. Zašto?

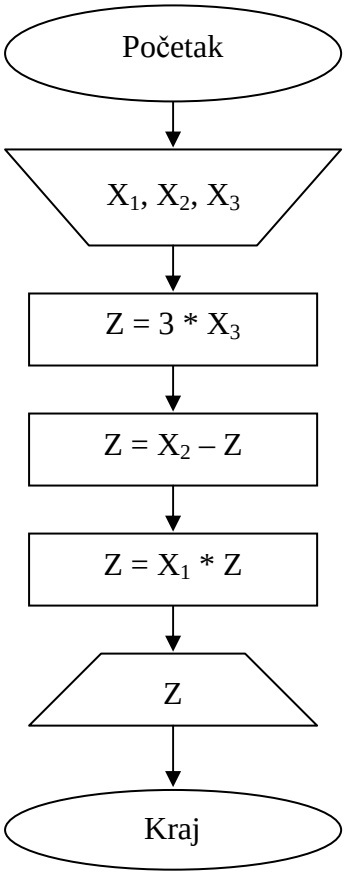
Rekli smo da je za optimalnost algoritma bitan utrošak memorije. A koliki je utrošak memorije zavisi od broja promjenljivih koje koristi algoritam.

U zadatku smo koristili promjenljive x_1, x_2, x_3, z_1, z_2, z i zadatak je izgledao:

Rešenje:



A mogao je da izgleda:



Ulazne veličine su X_1, X_2, X_3
Međurezultati veličine su Z_1, Z_2, Z

Vrijeme potrebno za izvršavanje algoritma naziva se vremenska složenost algoritma. Najčešće zavisi od broja operacija.

Optimalan je onaj algoritam koji ima najmanju vremensku složenost.

Primjer:

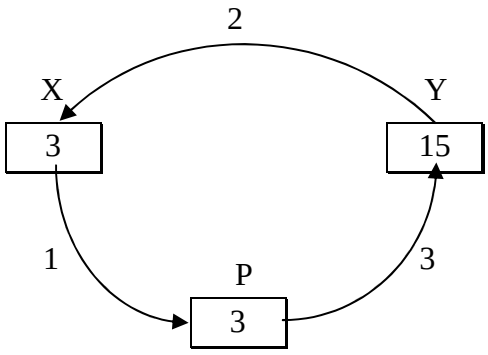
Prvi zadatak koji smo radili na času, a u kom je trebalo sastaviti algoritam za množenje prirodnih brojeva, pri čemu se koristi operacija sabiranja. Zadatak je bio tačno urađen, ali nije bio optimalan.

Vremenski nije isto uraditi:

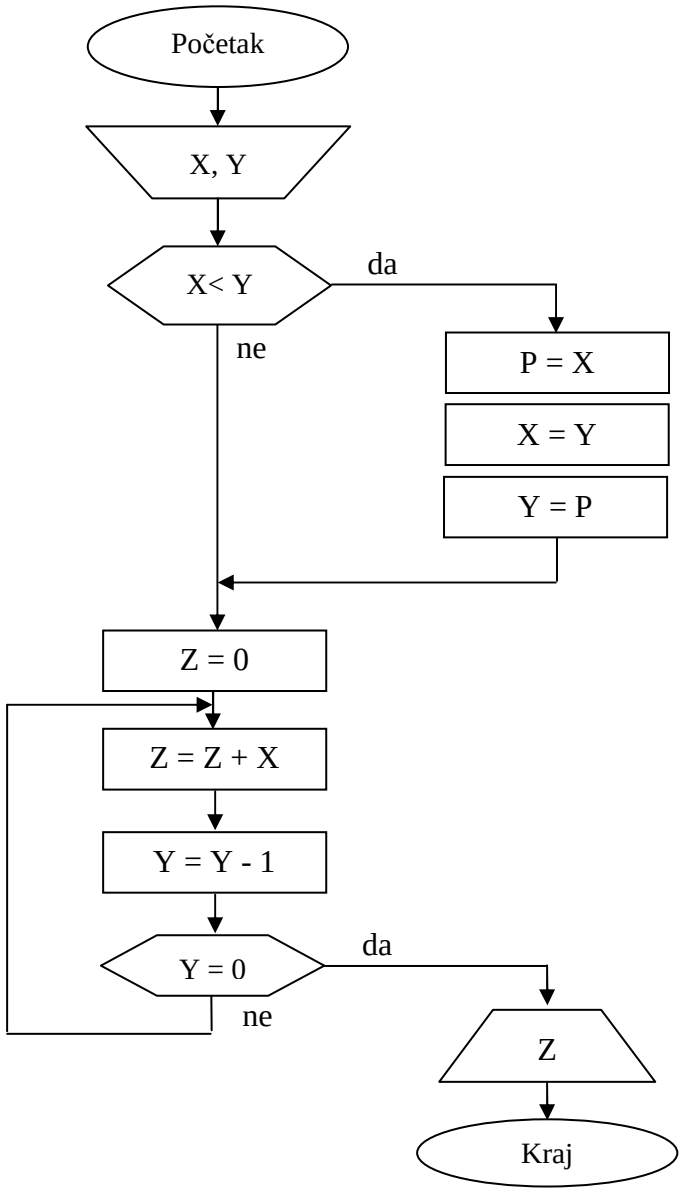
3 * 15 = 15 + 15 + 15 (3 operacije) ili

15 * 3 = 3 +3 + 3 + ... 3 (15 operacija)

Optimalno rešenje bi moglo biti:



Pomoćna kutija



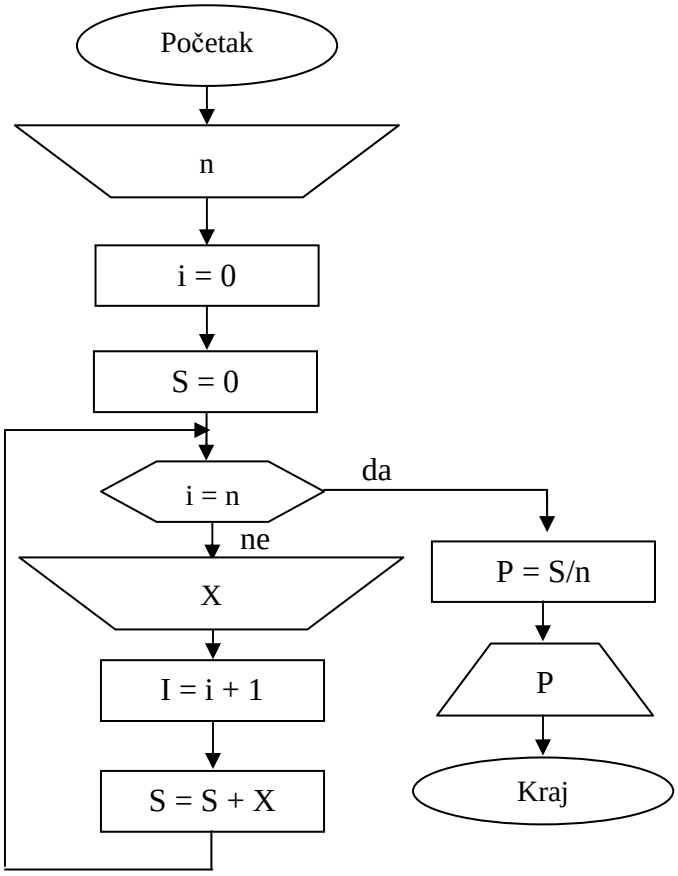
Optimalan je onaj algoritam koji ima minimalnu i prostornu i vremensku složenost.

Zadaci:

Naći optimalno rješenje za zadatke:

1. Sastaviti algoritamsku šemu za n>0 i x_1, x_2, x_3, … x_n koja izračunava prosječnu vrijednost P = (x_1 + x_2 + x_3 + … x_n) /n

Optimalno rješenje je:

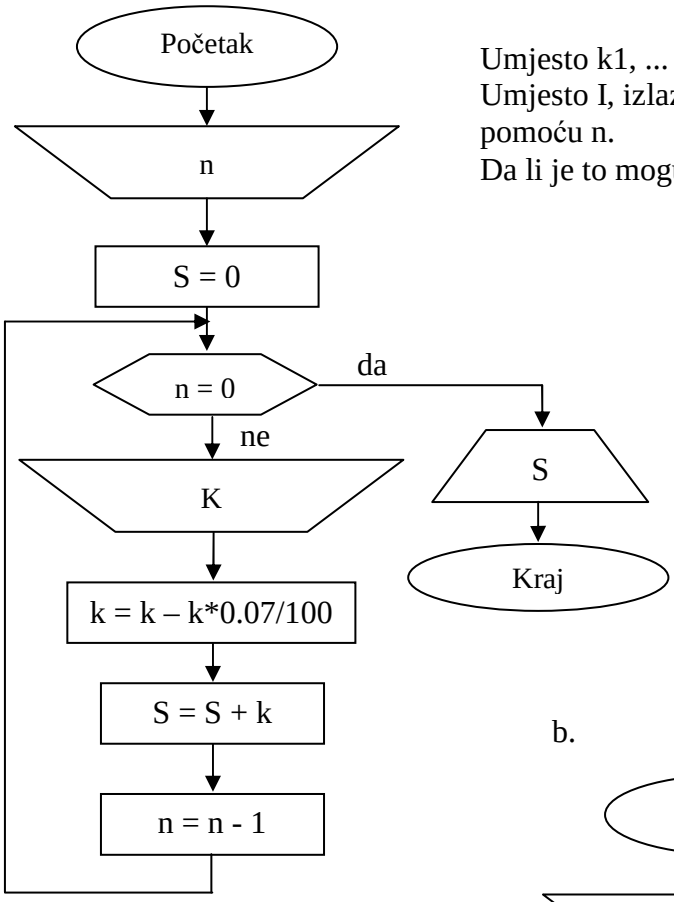


Umjesto x_1, x_2, … x_n koristimo samo promjenljivu x.

Sastaviti algoritamsku šemu koja za n i iznose novca k1, k2, k3, … kn izračunava:

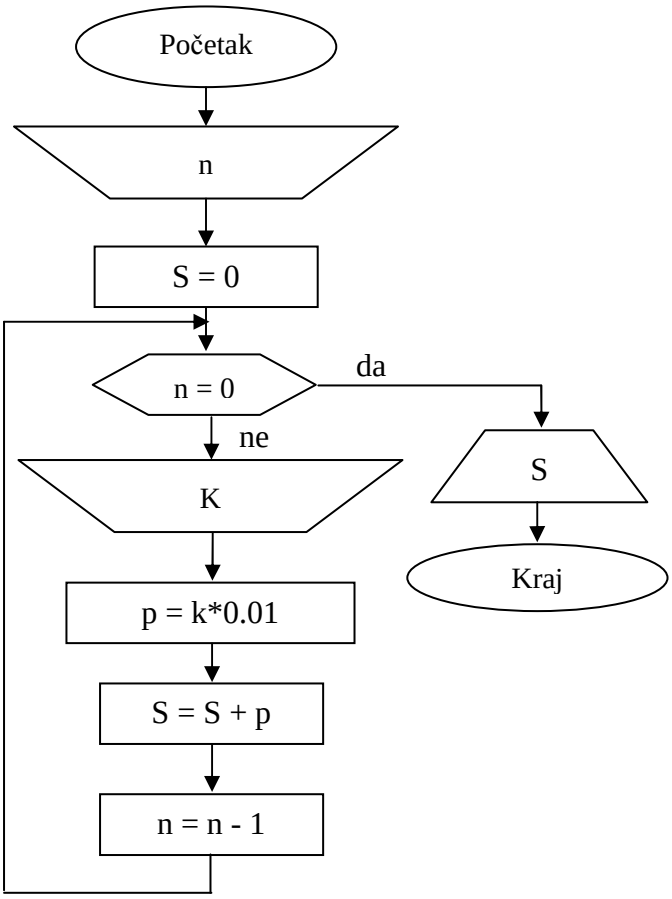
a. Ukupan iznos novca posle umanjenja svih iznosa za 0.07%
b. Ukupno povećanje svih iznosa na ime povećanja svakog pojedinačnog iznosa od 10%

a.



Umjesto k1, ... kn koristimo k.
Umjesto I, izlazni kriterijum ciklusa definišemo pomoću n.
Da li je to moguće kod prethodnog primjera. Zašto?

b.

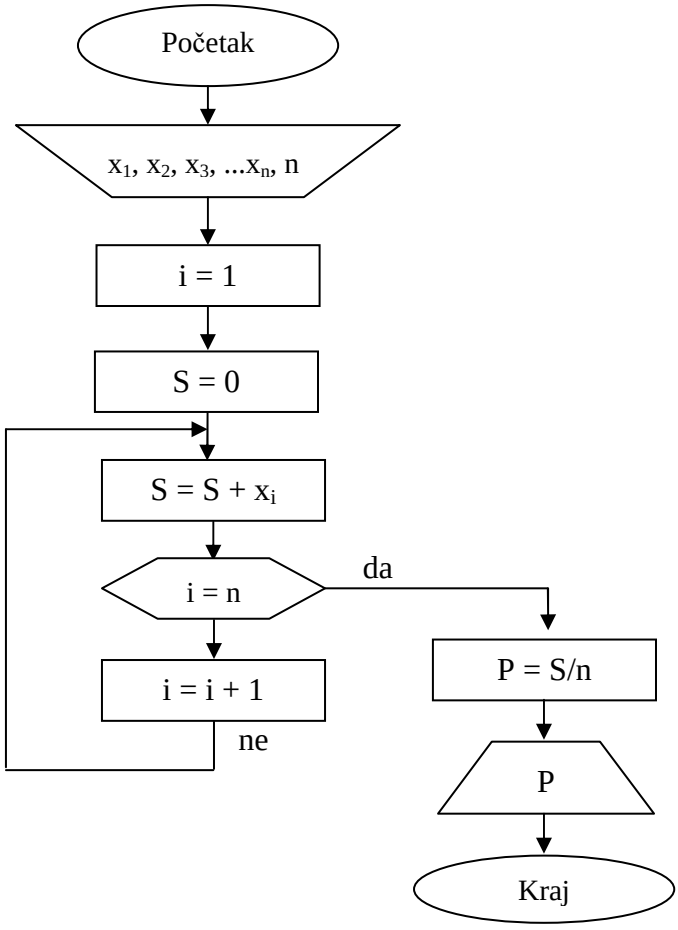


PROVJERA ISPRAVNOSTI ALGORITMA

Proces stvaranja algoritma je kreativan proces i nemoguće ga je formalizovati. Stoga je ovaj posao podložan greškama. Ovo je veliki problem, jer se greške u algoritmu kasnije odražavaju na program. Danas postoji veliki broj metoda kojim se dokazuje ispravnost algoritma.

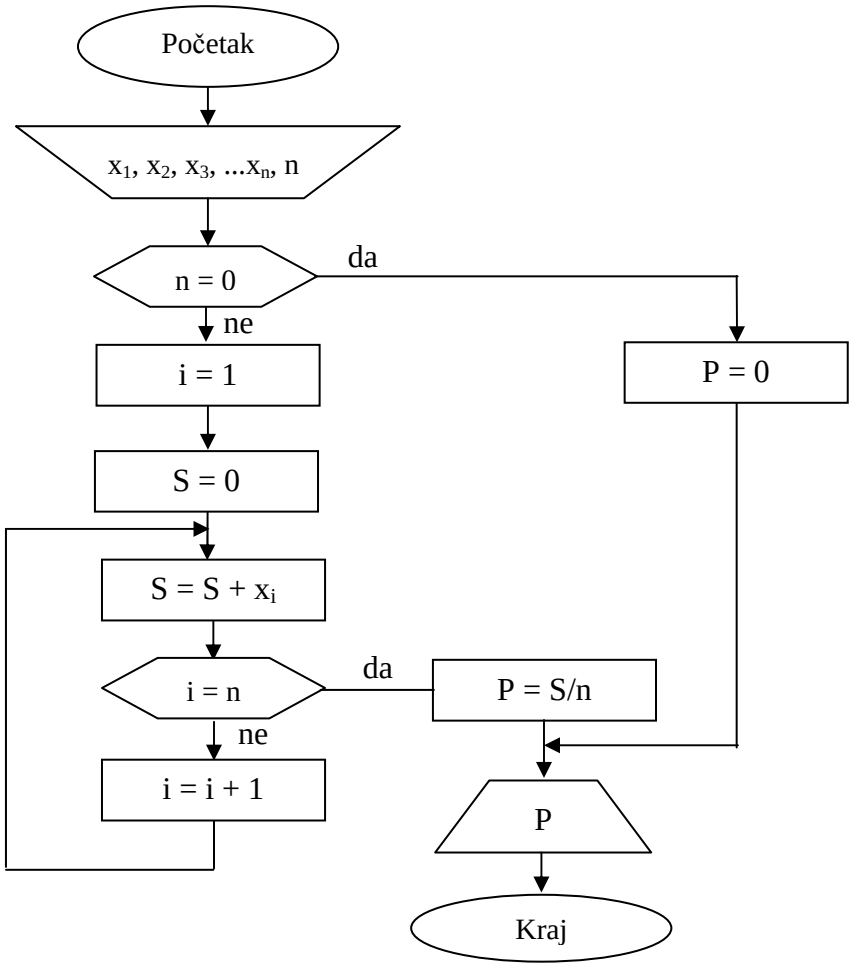
Umjesto dokazivanja ispravnosti algoritma, može se vršiti njegovo testiranje. Testiranje je ustvari provjera algoritma preko primjera. Testiranje nije uvek 100% sigurno (jer se i u procesu testiranja mogu potkrati greške), ali se sa velikom vjerovatnoćom mogu otkriti greške.

Možemo obaviti testiranje na već urađenom primeru, koji je računao prosek n brojeva (Primjer 2a*) i koji je izgledao ovako:



Ako je n = 2 i $x_1 = 1$ i $x_2 = 3$
Testiramo korak po korak i ako rezultat bude 2, onda je i algoritam tačan.

Ako je n=0 onda ovaj algoritam nije ispravan. Tada vršimo njegovu korekciju, pa bi algoritam izgledao:



Algoritam Primjer 2a* je ispravan za n>0, ali zbog osobine 5 (masovnost) treba težiti da on važi za za najširi skup ulaznih veličina, pa i za n=0.

OSOBINE ALGORITAMA

I pored raznovrsnosti, algoritmi imaju neke zajedničke osobine:

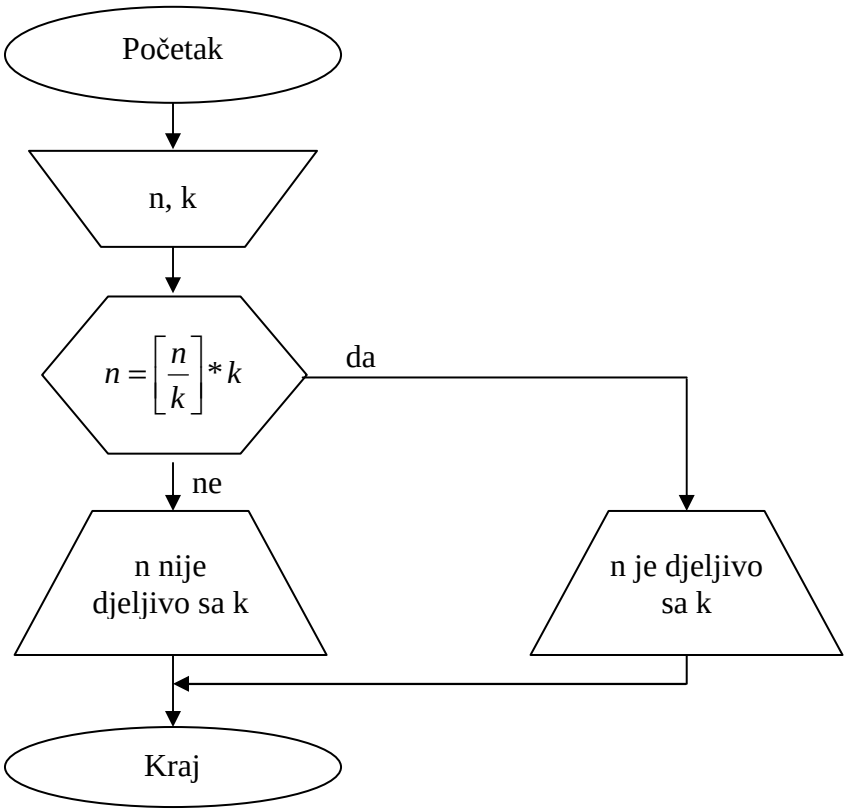
1) ***Diskretnost algoritma.*** Ako posmatramo izvršavanje algoritama u vremenu, tada svakom algoritamskom koraku možemo pridružiti diskretan vremenski period u kom se taj korak izvršava.
2) ***Determinisanost algoritma.*** Svaki algoritamski korak sadrži ulazne veličine na osnovu kojih se jednoznačno određuju izlazne veličine.
3) ***Elementarnost algoritamskih koraka.*** Zakon dobijanja izlaznih veličina na osnovu ulaznih mora biti jasan i prost.
4) ***Rezultativnost algoritma.*** Za svaki mogući skup ulaznih veličina u algoritmu mora biti definisano šta se smatra rezultantom.
5) ***Masovnost.*** Algoritam treba uraditi tako da važi za najširi skup ulaznih veličina.

ZADACI ZA VJEŽBU

1. Sastaviti algoritam koji ispituje da li je prirodan broj N djeljiv prirodnim brojem K

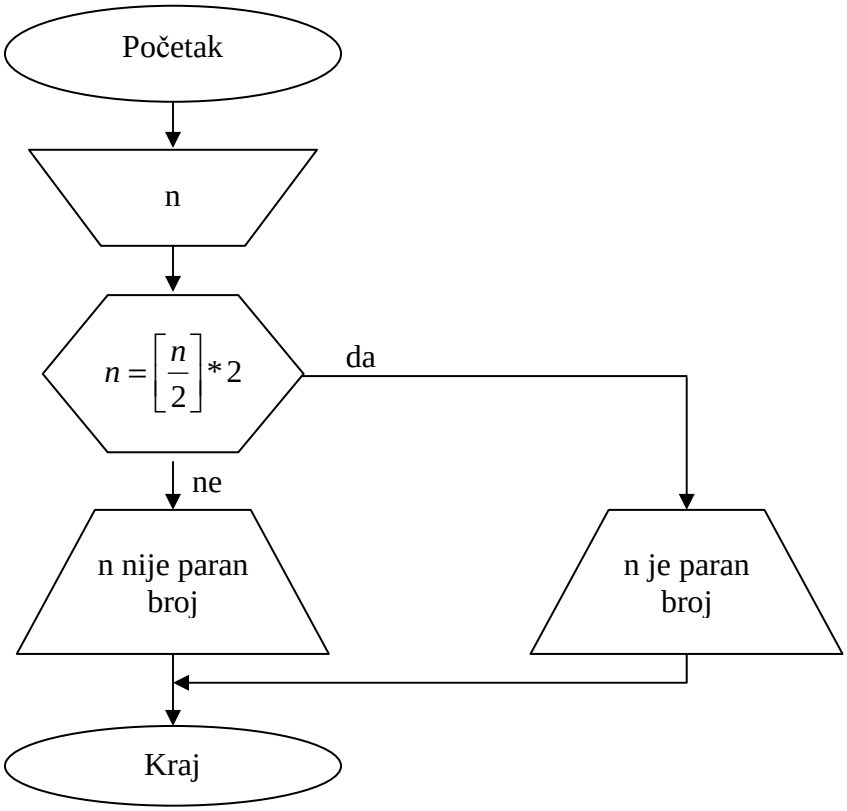
$$n = \left[\frac{n}{k}\right] * k \qquad 15 = \left[\frac{15}{3}\right] * 3$$

Uzimaju se u obzir cjelobrojne vrijednosti.



2. Sastaviti algoritam koji ispituje da li je prirodan broj n paran

$$n = \left[\frac{n}{2}\right] * 2$$



3. Sastaviti algoritam koji nalazi zbir cifara prirodnog broja n

Primjer: Za broj 2431 naći zbir cifara

2 4 3 1

2431 - $\left(\frac{2431}{10}\right)$*10 = 2431 - 2430 = 1

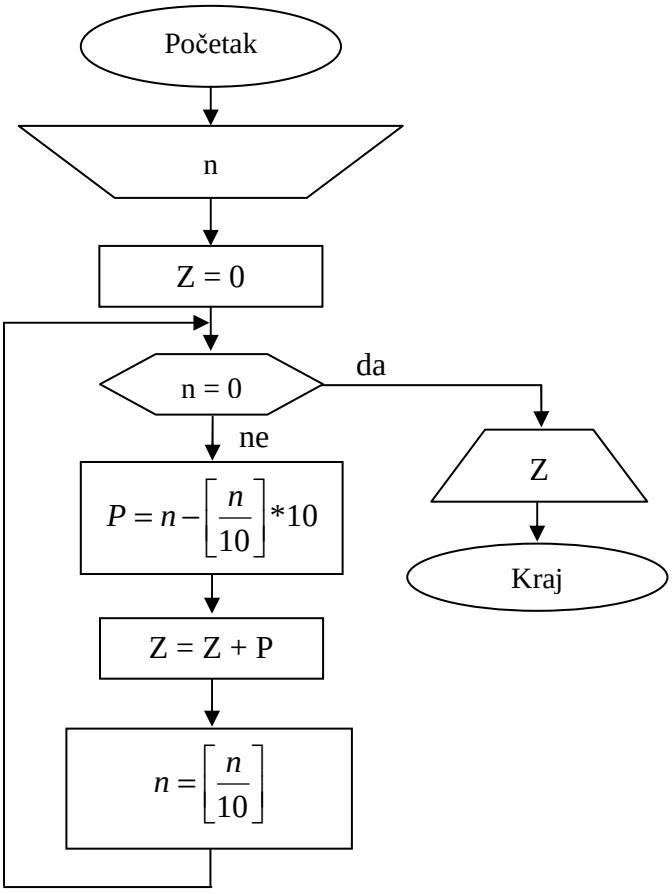
243 - $\left(\frac{243}{10}\right)$*10 = 243 - 240 = 3

24 - $\left(\frac{24}{10}\right)$*10 = 24 - 20 = 4

2 - 0 = 2

1 + 3 + 4 + 2 = 10

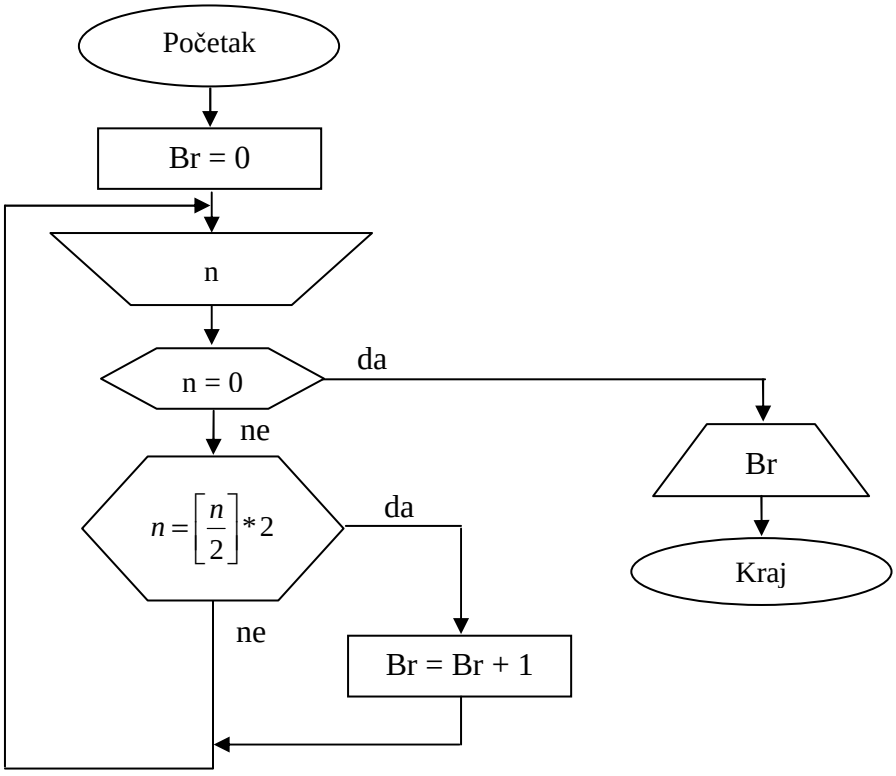
$$P = n - \left[\frac{n}{10}\right] * 10$$



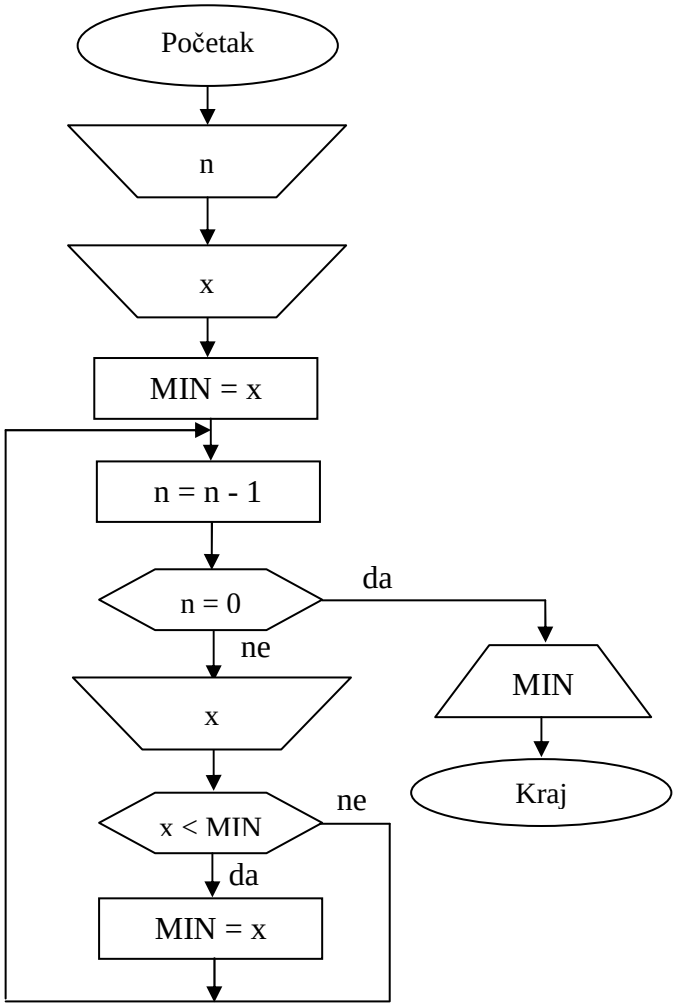
4. Unose se prirodni brojevi n sve dok se ne unese nula. Sastaviti algoritam koji će izbrojati koliko ukupno ima parnih brojeva.

Paran broj je:

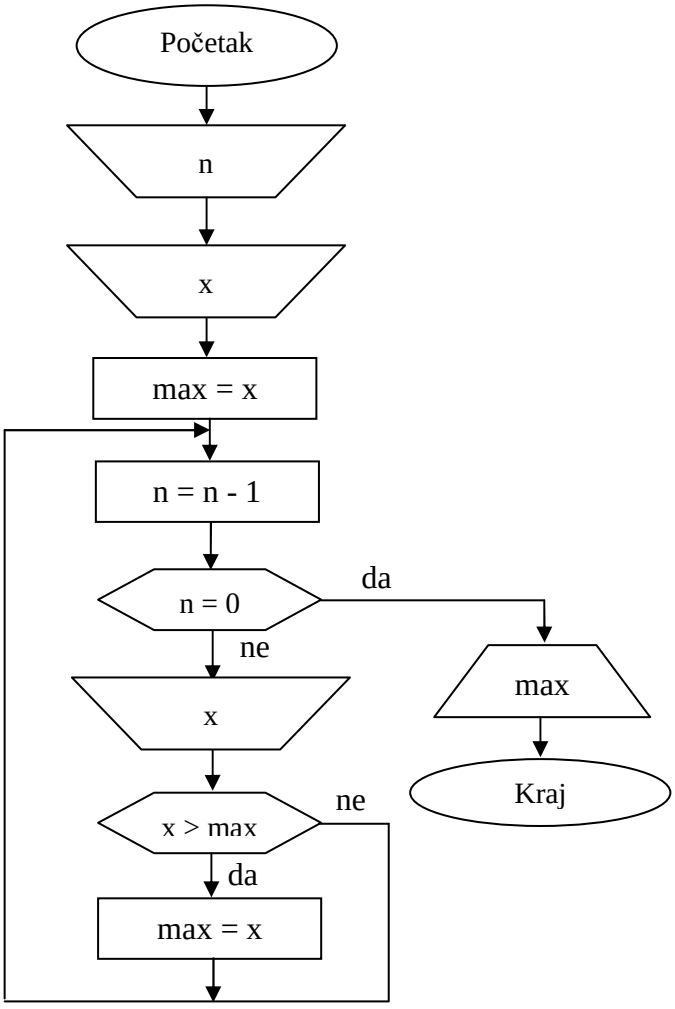
$$n = \left[\frac{n}{2}\right] * 2$$



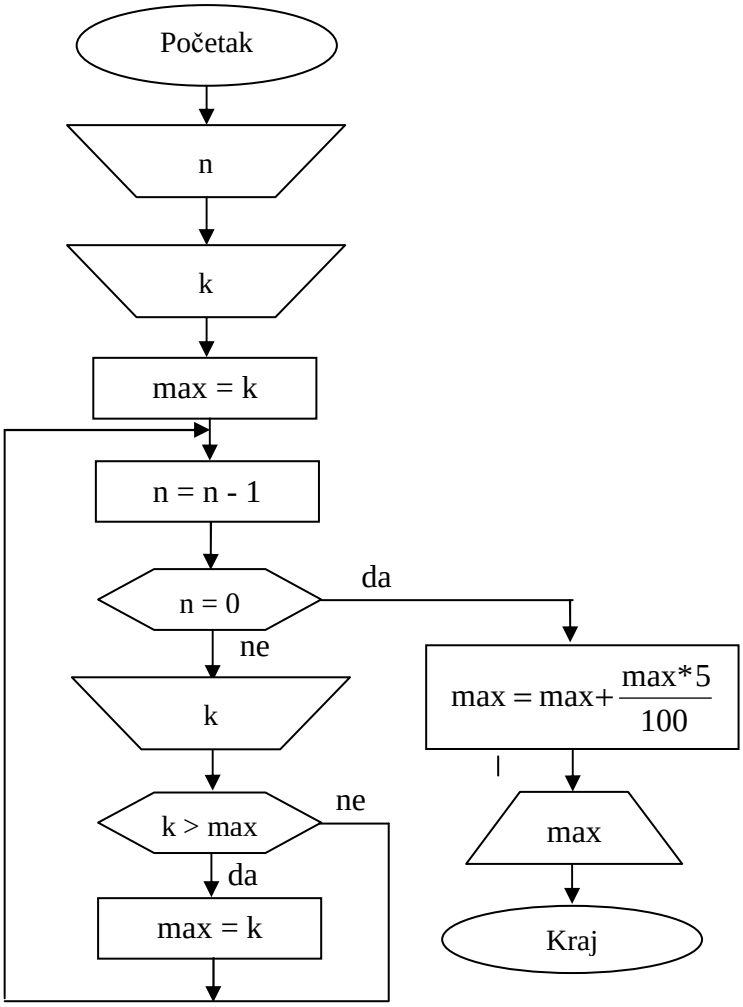
5. Unose se n>0 i n prirodnih brojeva. Sastaviti algoritam koji određuje najmanji među njima.



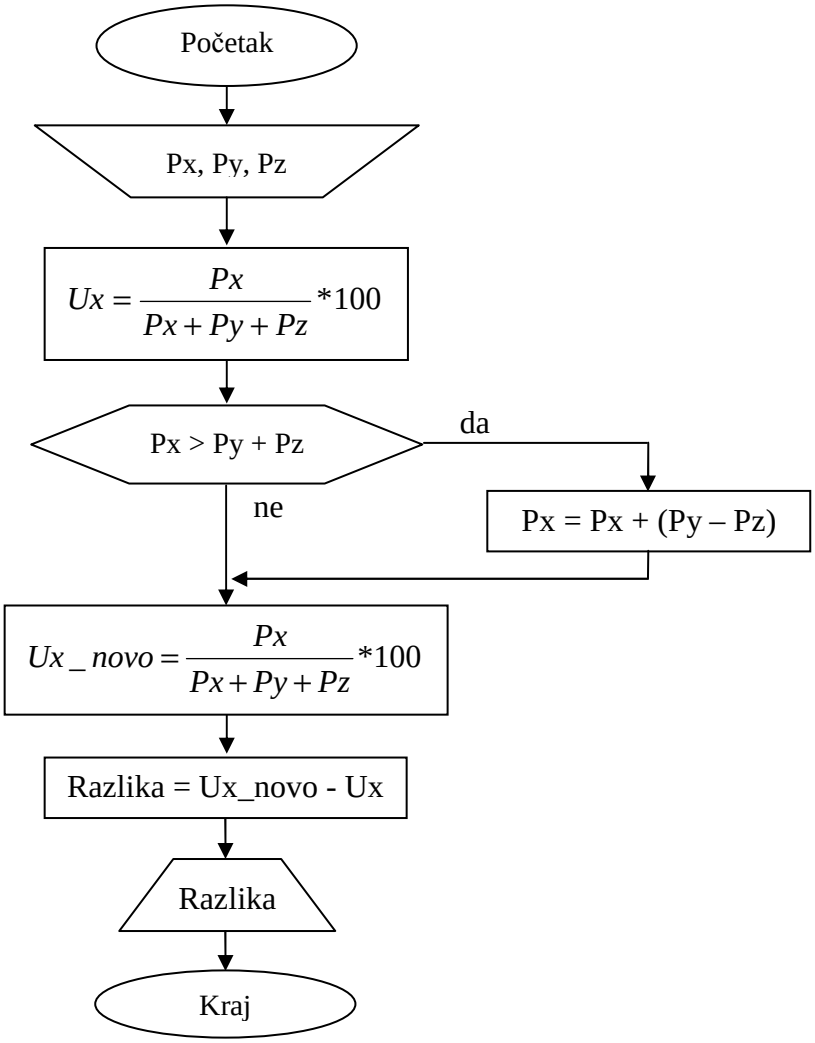
6. Učitava se prirodan broj n (n>0) i n prirodnih brojeva. Sastaviti algoritam koji određuje najveći među njima.



7. Na depozitne račune kod banke sliva se n štednih uloga građana. Sastaviti algoritam koji najveći uloženi iznos uvećava za 5%.



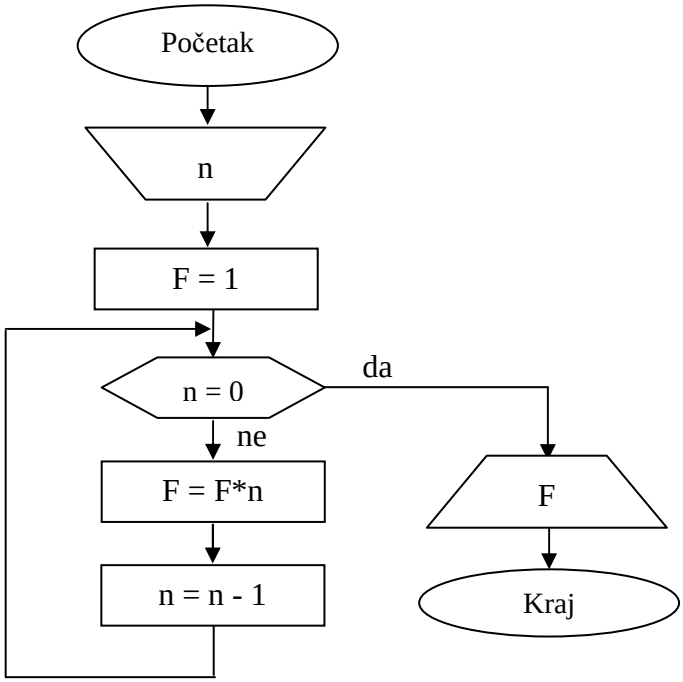
8. Unose se prihodi preduzeća X, Y, Z. Ukoliko je prihod prvog preduzeća veći od zbira prihoda druga dva preduzeća, onda se on povećava za iznos razlike prihoda preduzeća Y i Z. Sastaviti algoritam koji izračunava za koliko procenata će se promijeniti prihodovno učešće preduzeća X u ukupnom prihodu ova tri preduzeća.



Ux – učešće prihoda preduzeća X u ukupnom prihodu
Ux_novo – promijenjeno – novo učešće

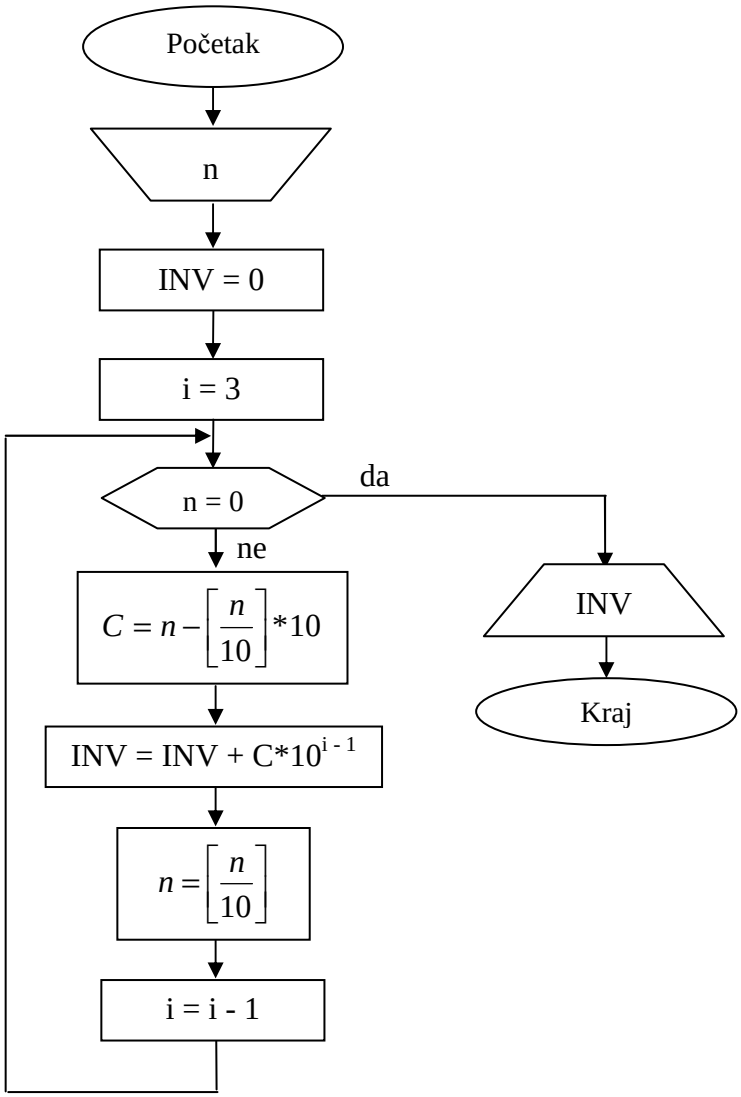
9. Sastaviti algoritam koji računa n!

n! = n*(n-1)*(n-2)...3*2*1
0! = 1



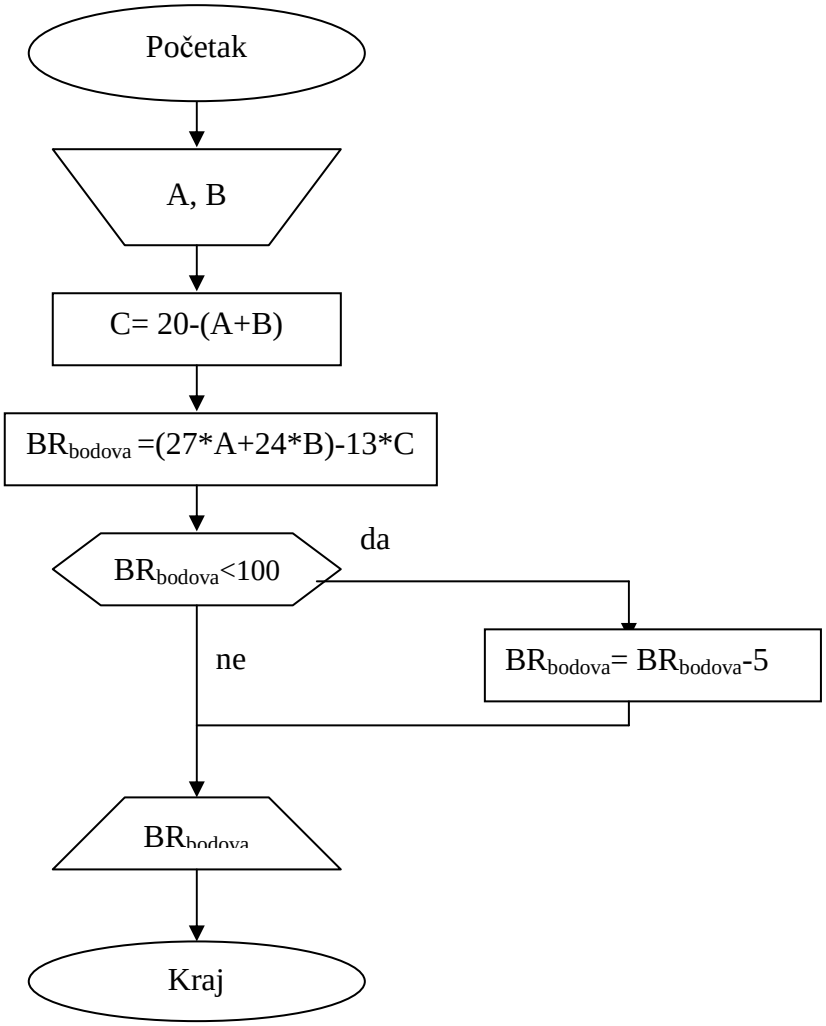
10. Unosi se trocifren broj n. Sastaviti algoritam koji pronalazi broj inverzan broju n (sa obrnutim redosljedom cifara).

    123 → 321



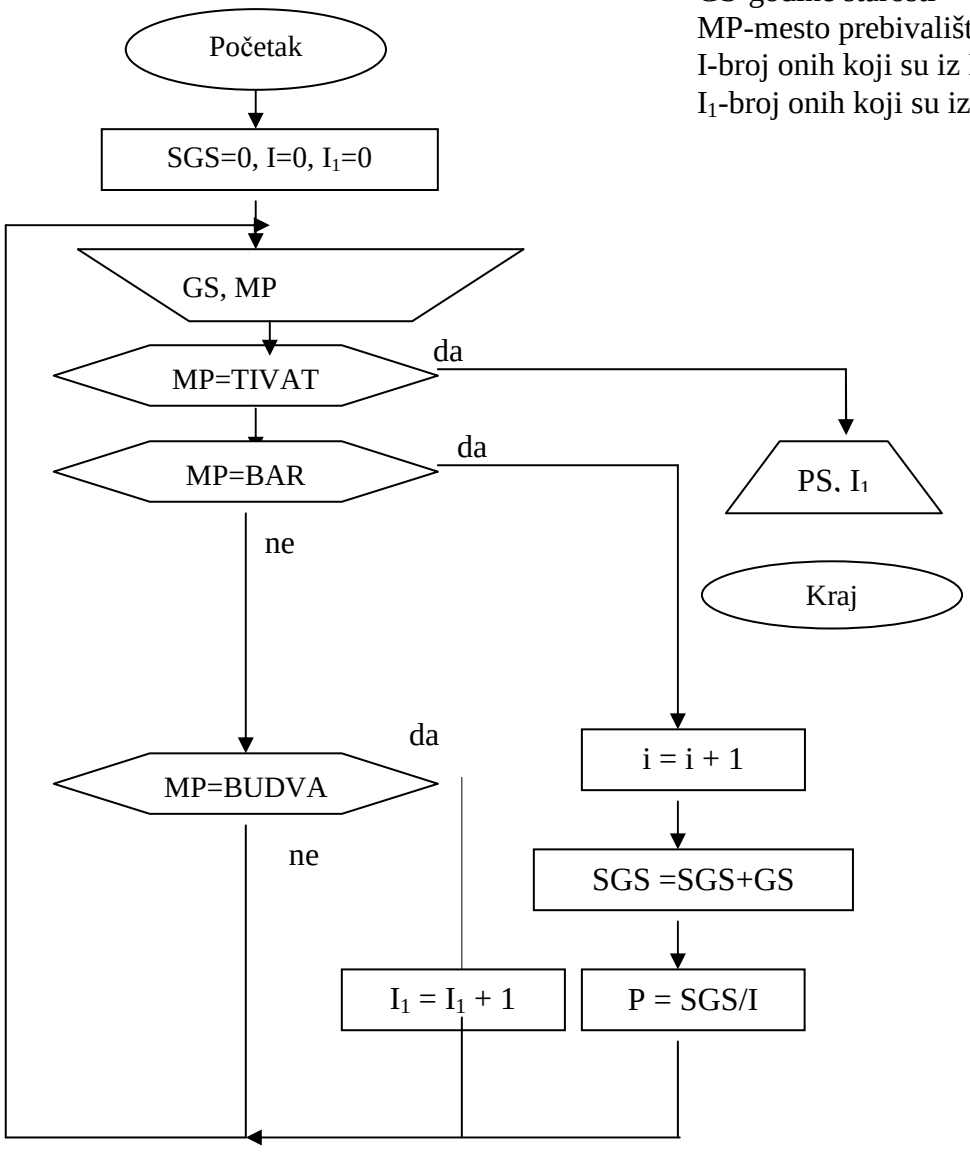
11. Od 40 pitanja, od kojih je 20 težih i 20 lakših, studenti odgovaraju na 20 pitanja. Tačan odgovor na teže pitanje nosi 27 bodova, a na lakše 24 boda. Za svaki netačan odgovor studentu se oduzima 13 bodova. Ukoliko je ukupan broj bodova veći od 100, algoritam treba da prikaže osvojeni broj bodova, ukoliko uslov nije ispunjen prikazuje se broj bodova umanjen za 5.

A – broj tačnih odgovora na teža pitanja
B – broj tačnih odgovora na lakša pitanja
C– broj netačnih odgovora C=20-(A+B)

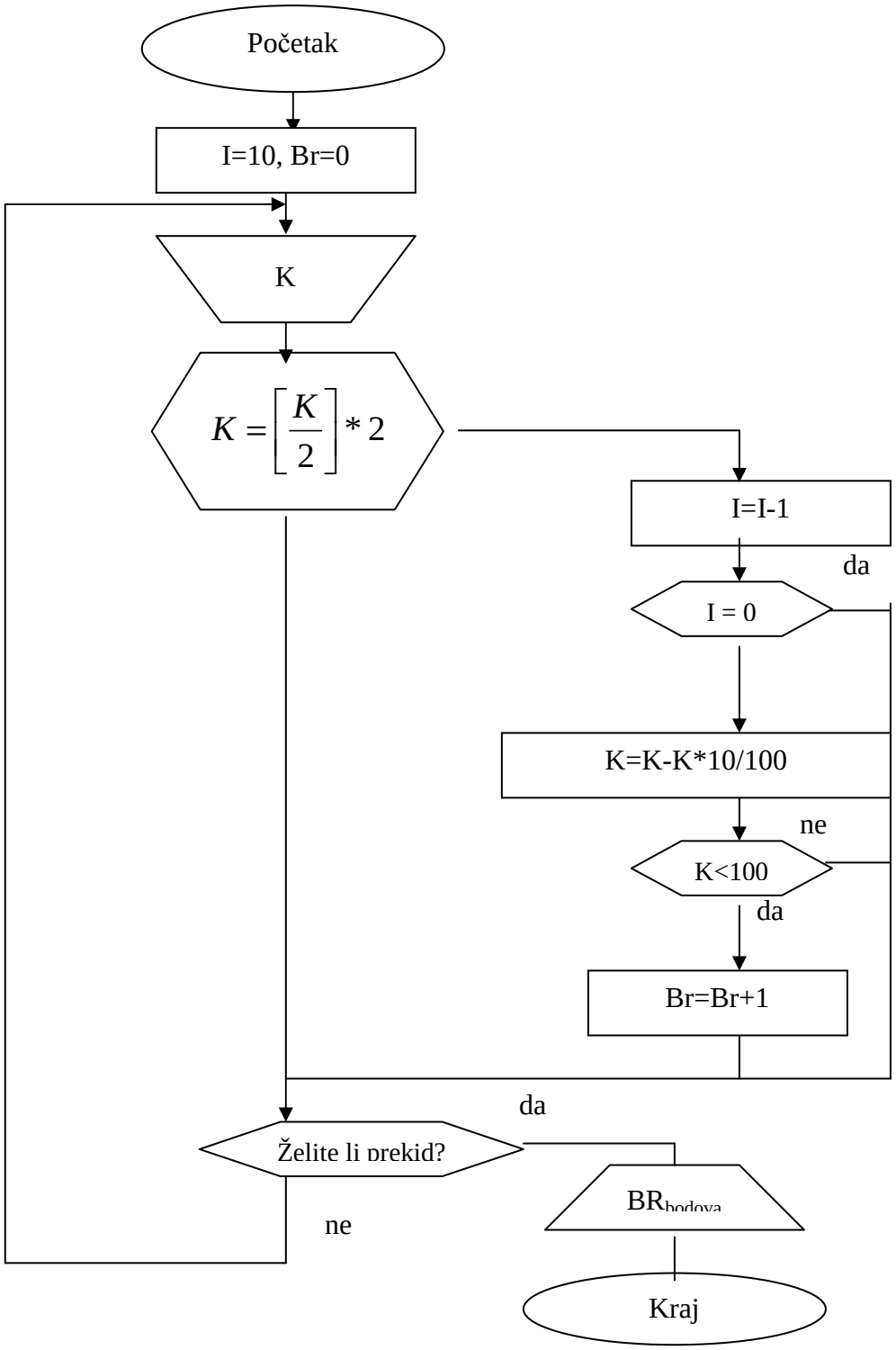


12. Unose se podaci o godinama starosti i mestu prebivališta za građane primorskih opština, sve dok se ne pojavi podatak TIVAT. Sastaviti algoritam koji računa prosečnu starost u opštini Bar i na kraju prikazuje taj podatak i podatak koliki je broj građana u opštini Budva.

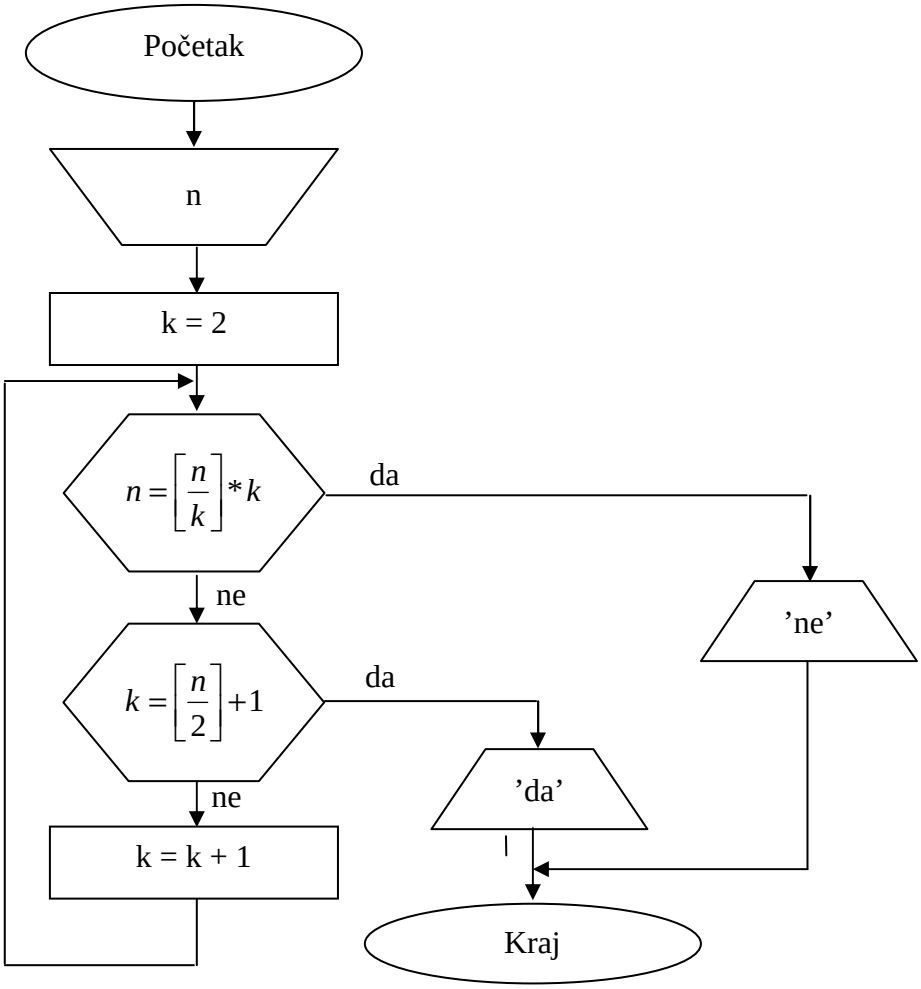
SGS-suma godina starosti
GS-godine starosti
MP-mesto prebivališta
I-broj onih koji su iz Bara
I_1-broj onih koji su iz Budve



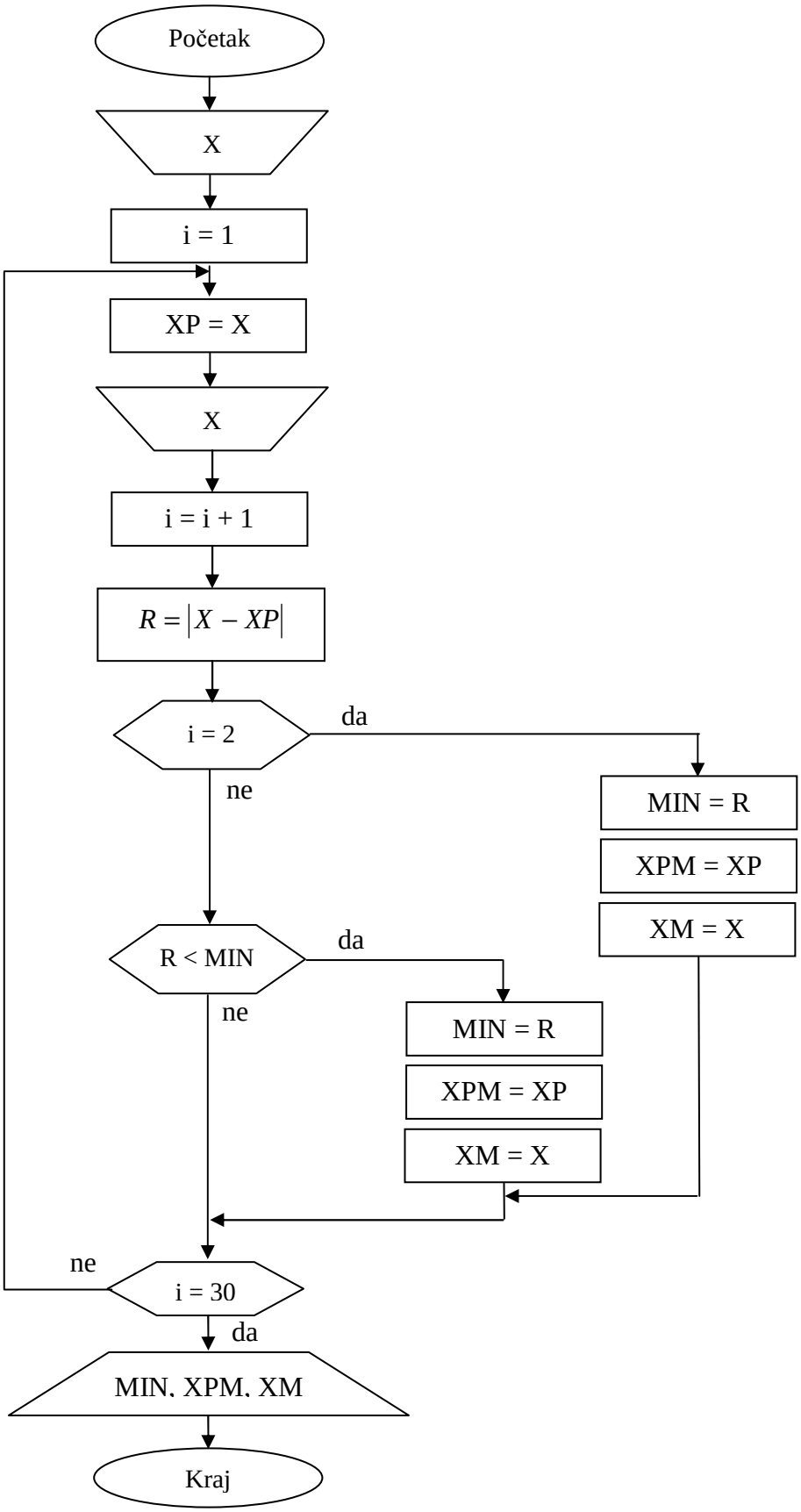
13. Unose se novčani iznosi sve dok se ne zatraži prekid. Sastaviti algoritam koji za prvih 10 parnih iznosa ispituje koliko je onih koji su posle umanjenja od 10% bili manji od 100. Na kraju prikazuje taj broj (Ukoliko se algoritam prekine pre 10-og parnog broja treba prikazati brojač iz uslova do trenutka prekida).



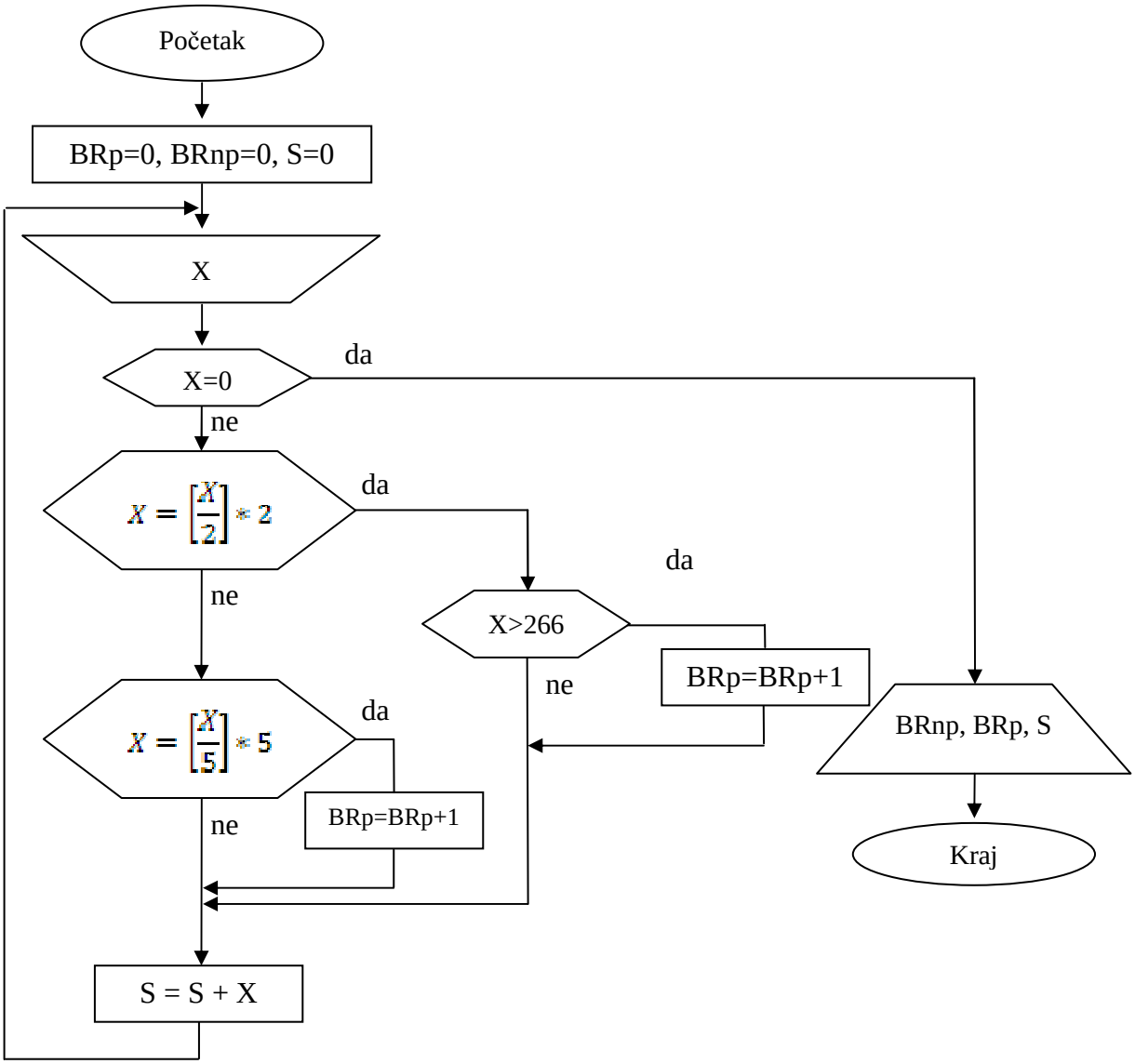
14. Sastaviti algoritam kojim se ispituje da li je prirodni broj n>1 prost? (djeljiv samo sa 1 i sa samim sobom)



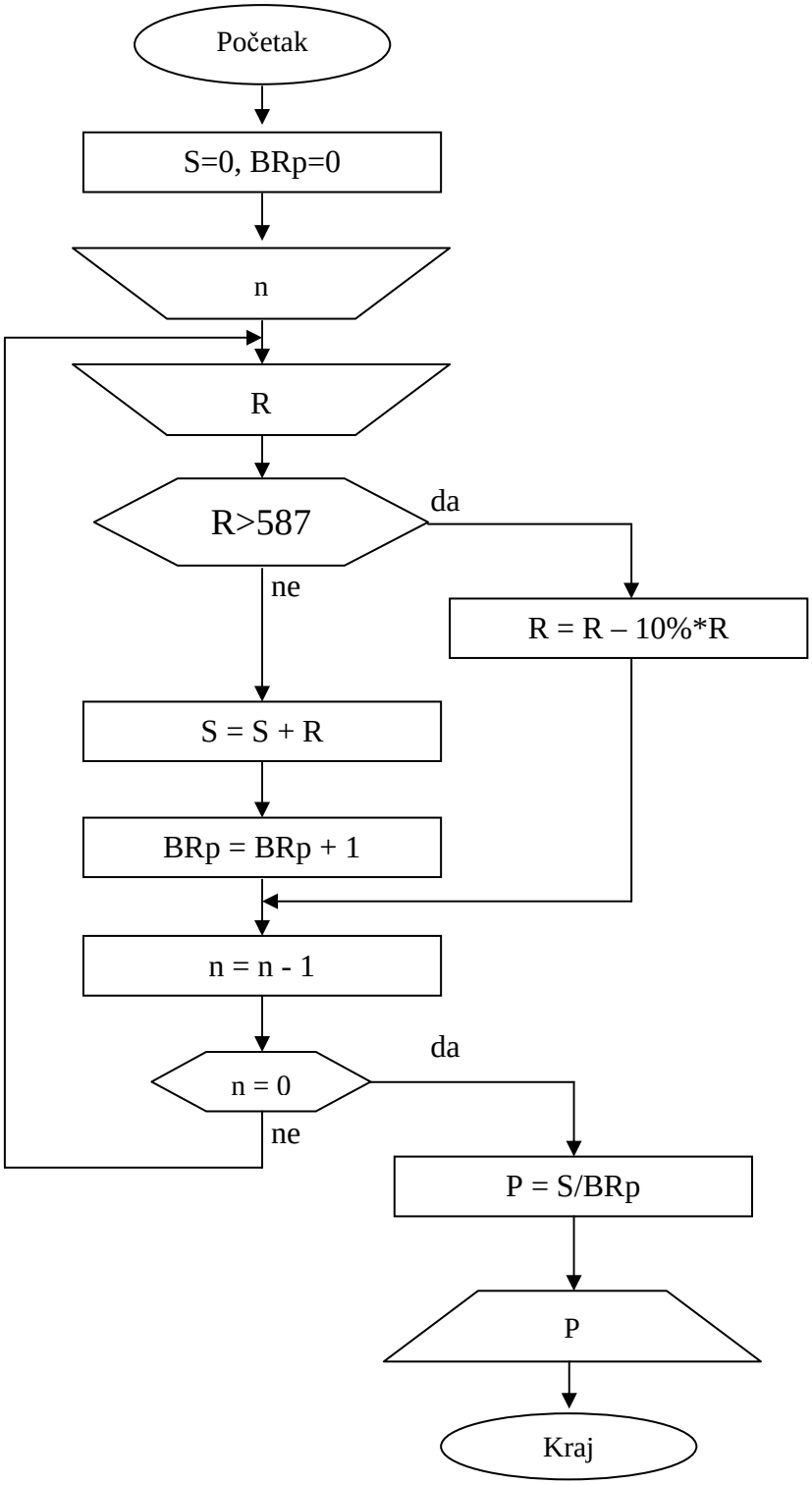
15. Učitava se 30 brojeva. Sastaviti algoritam koji pronalazi 2 uzastopna broja čija je apsolutna razlika najmanja.



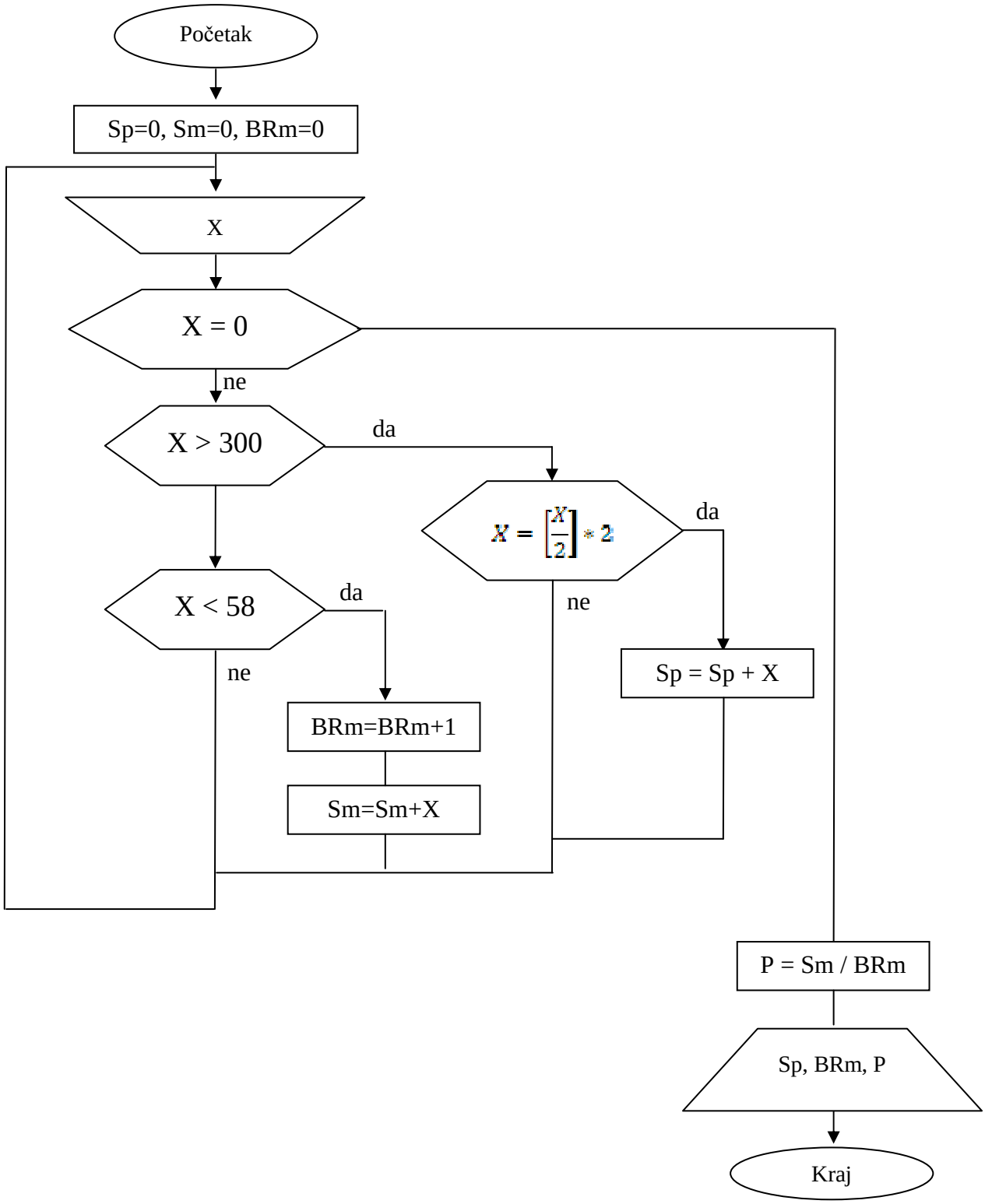
16. Unose se brojevi dok se ne unese nula. Sastaviti algoritam koji broji parne brojeve veće od 266, i neparne brojeve djeljive brojem 5 i računa sumu svih unesenih brojeva.



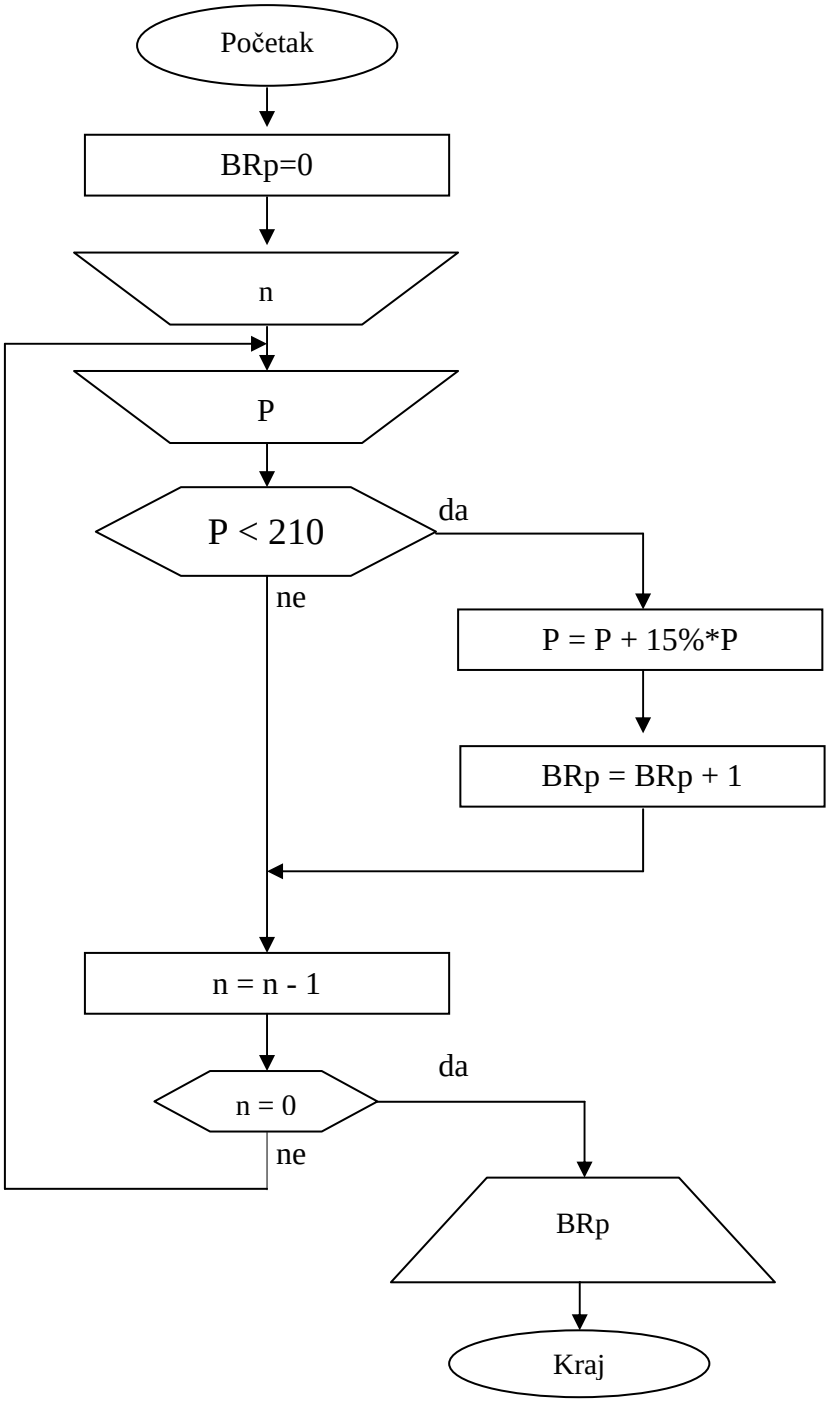
17. Unosi se n>0 rata za otplatu kredita. Rata koja je veća od 587 eura umanjuje se za 10%. Sastaviti algoritam koji računa prosjek rata kojima se ne umanjuje iznos.



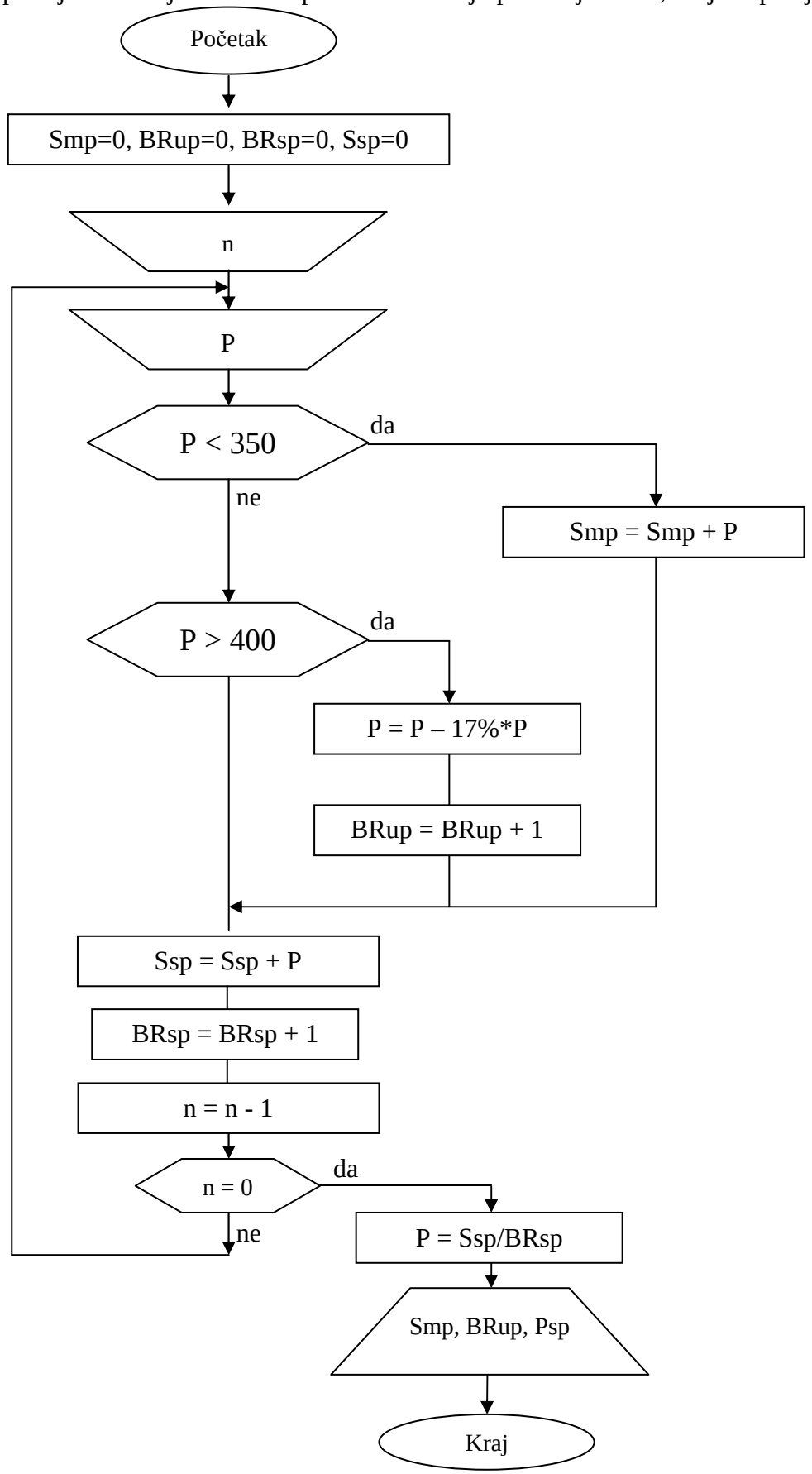
18. Unosi se prirodni brojevi dok se ne unese 0. Sastaviti algoritam koji računa sumu onih koji su veći od 300 i parni i prikazuje tu sumu, i broji i računa prosjek onih koji su manji od 58 i prikazuje broj i prosjek.



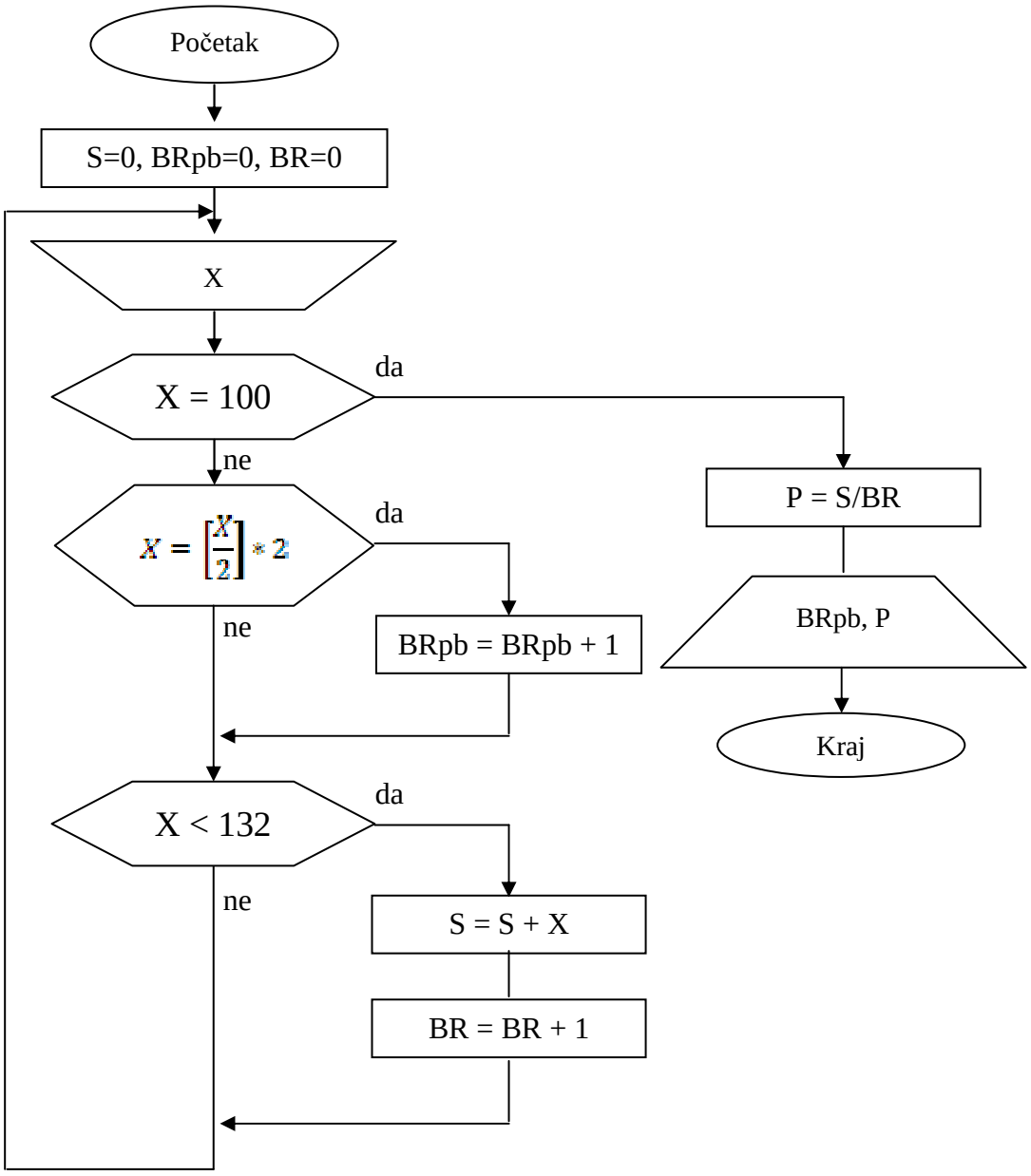
19. Unosi se n>0 plata zaposlenih. Sastaviti algoritam koji svaku platu koja je manja od 210 eura uvećava za 15% i broji koliko je tako uvećanih plata.



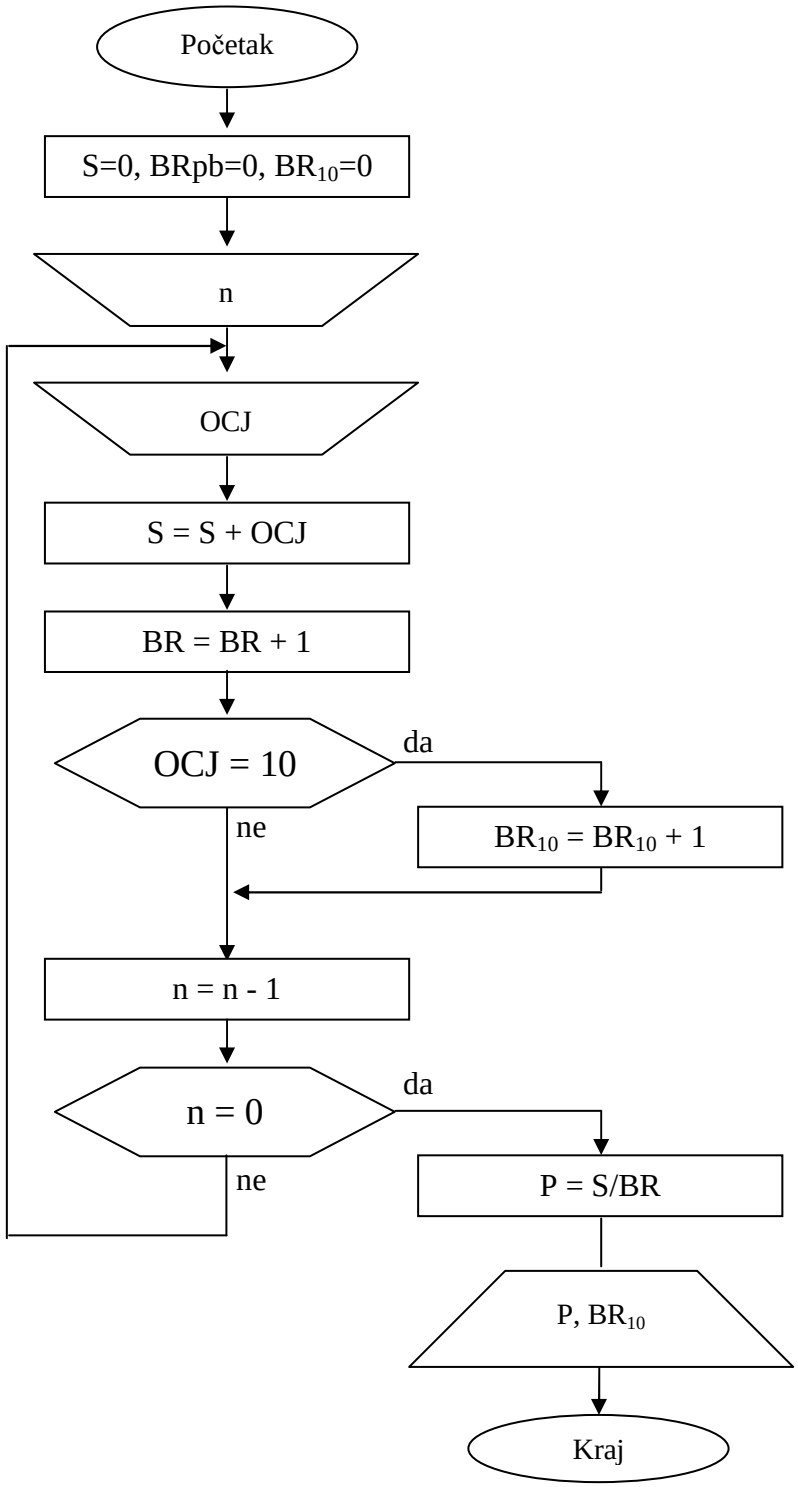
20. Unosi se n>0 plata zaposlenih. Sastaviti algoritam koji računa sumu plata manjih od 350 a plate koje su veće od 400 umanjuje za iznos poreza od 17% i broji koliko je tako umanjenih plata. Takođe, računa prosjek svih unešenih plata sa svim promjenama koje su bile na platama. Na kraju prikazuje sumu, brojač i prosjek.



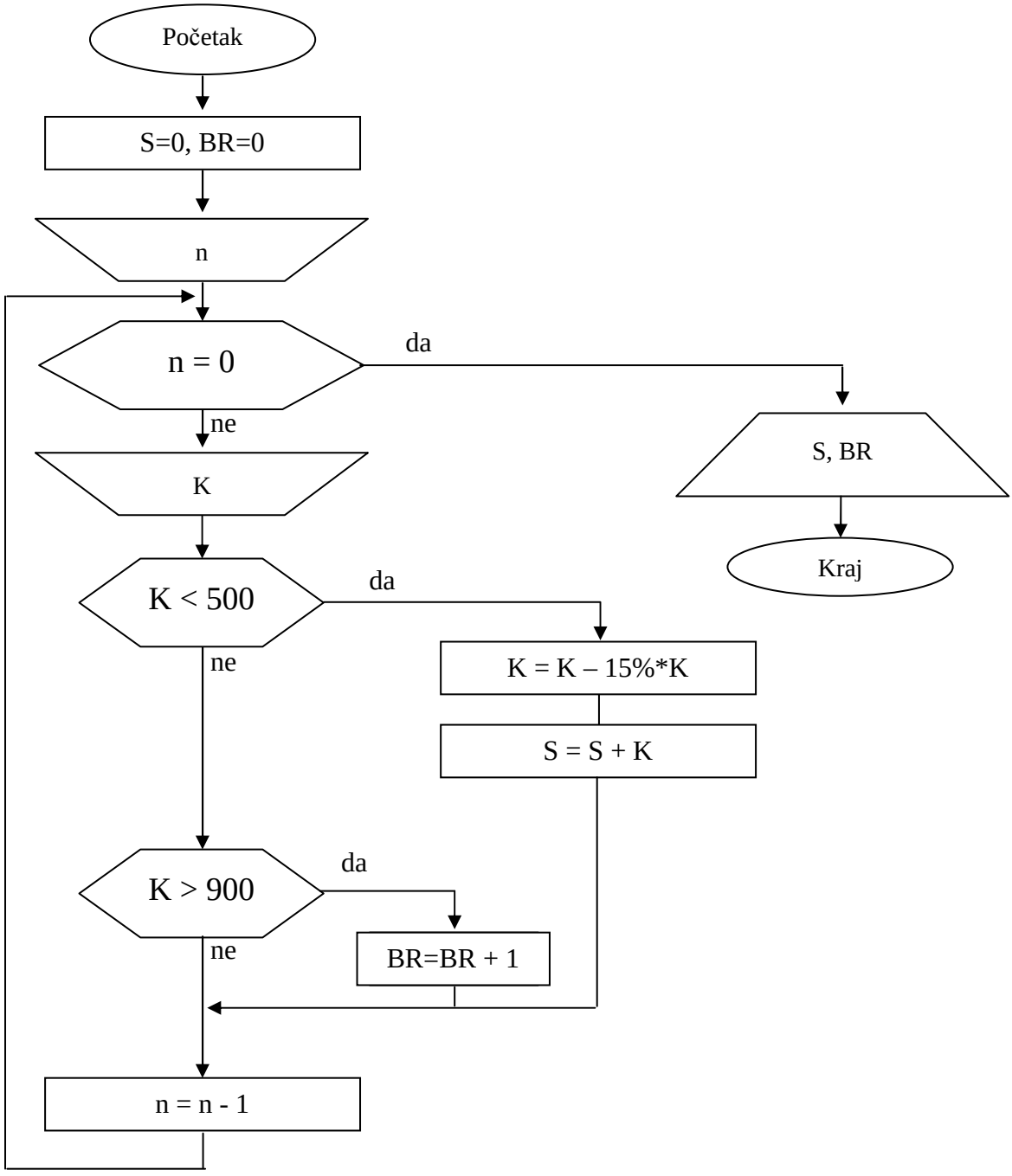
21. Unose se prirodni brojevi dok se ne unese broj 100. Sastaviti algoritam koji broji koliko je parnih brojeva, i računa prosjek svih brojeva manjih od 132.



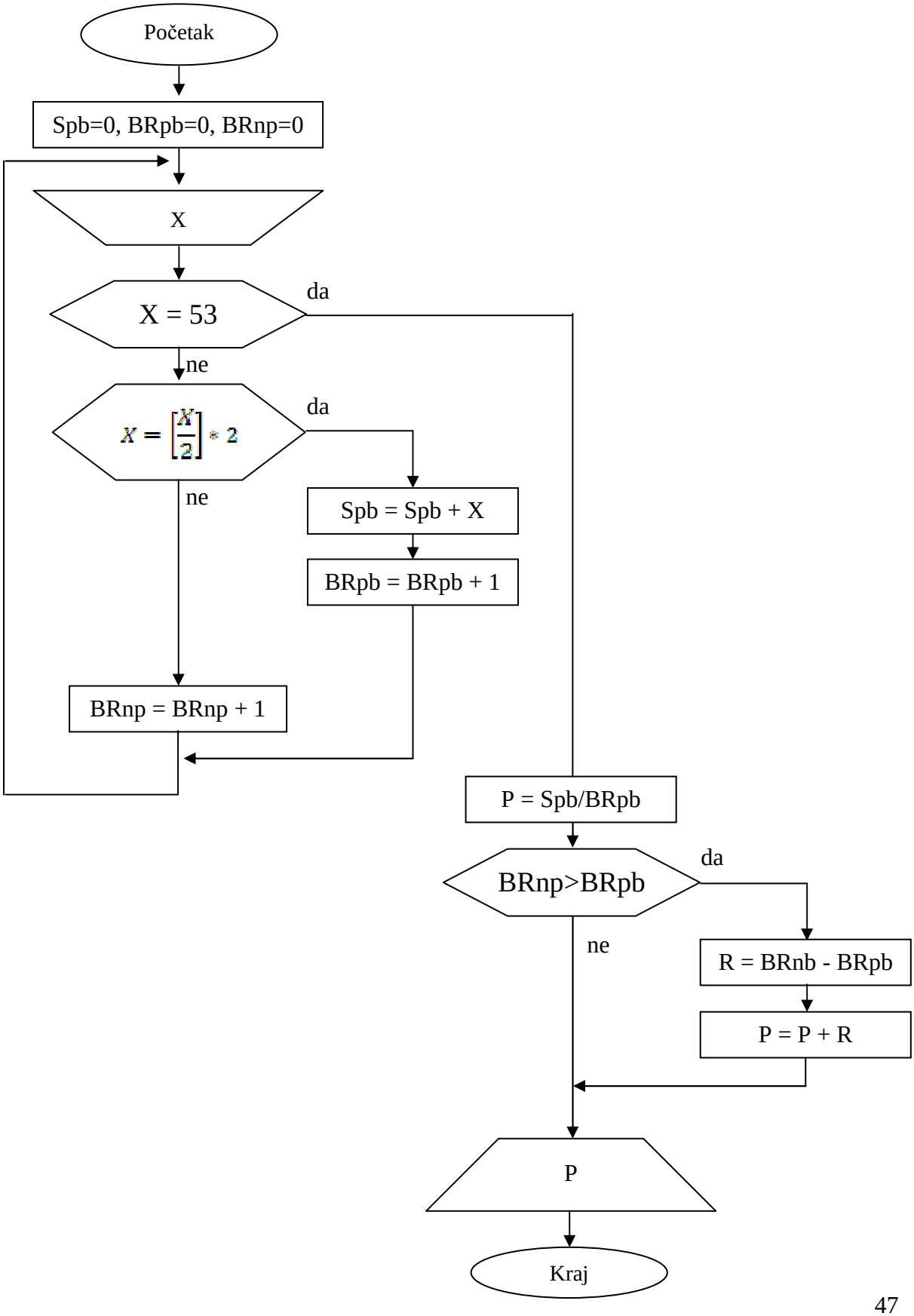
22. Unose se ocjene iz informatike za n>0 studenata. Sastaviti algoritam koji računa prosječnu ocjenu iz informatike kao i ukupan broj studenata koji su dobili ocjenu 10.



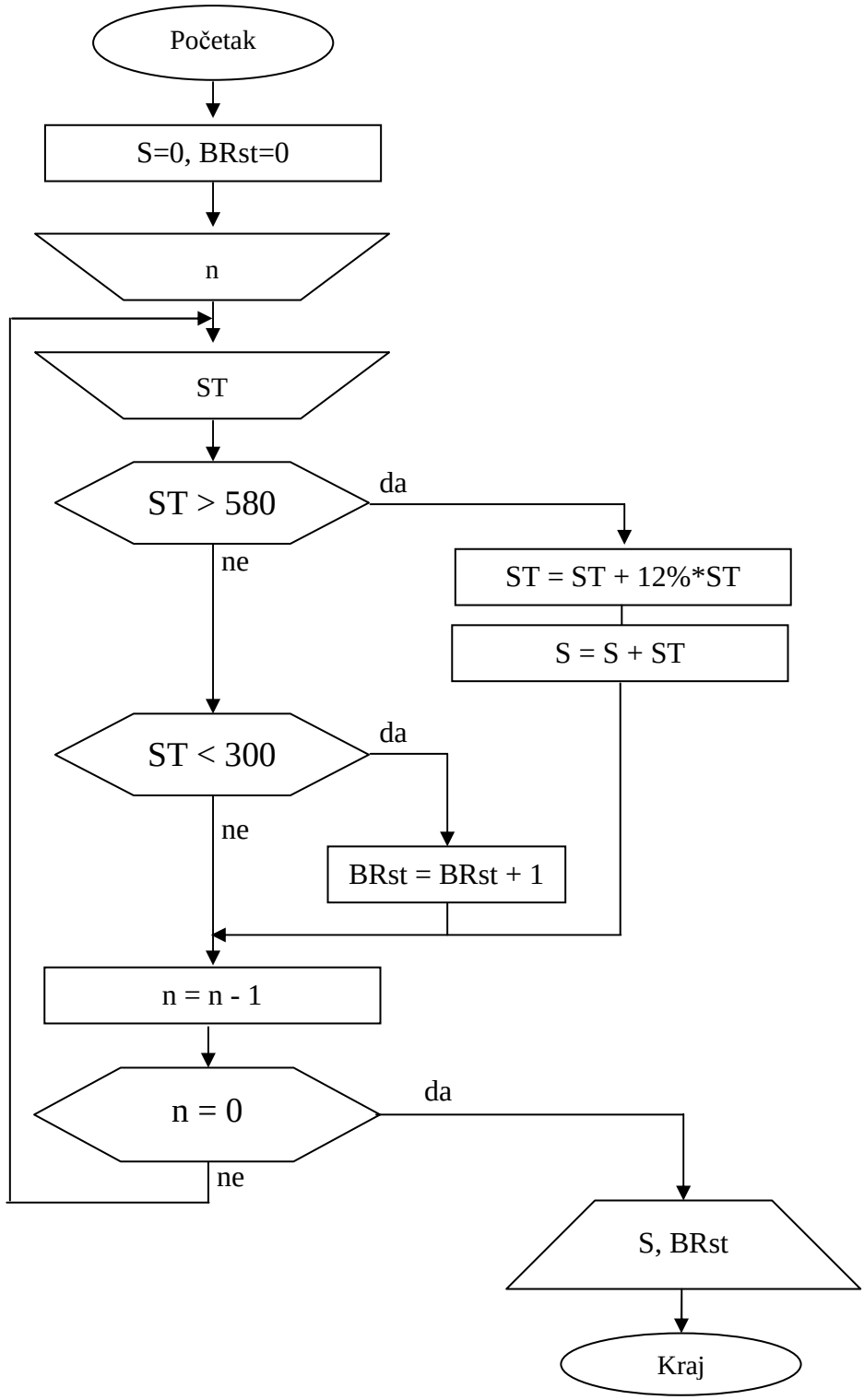
23. Unose se podaci o otplatama kredita (n>0). Sastaviti algoritam koji računa sumu svih iznosa koji su manji od 500 i umanjeni za 15%, i broji koliko je iznosa koji su veći od 900.



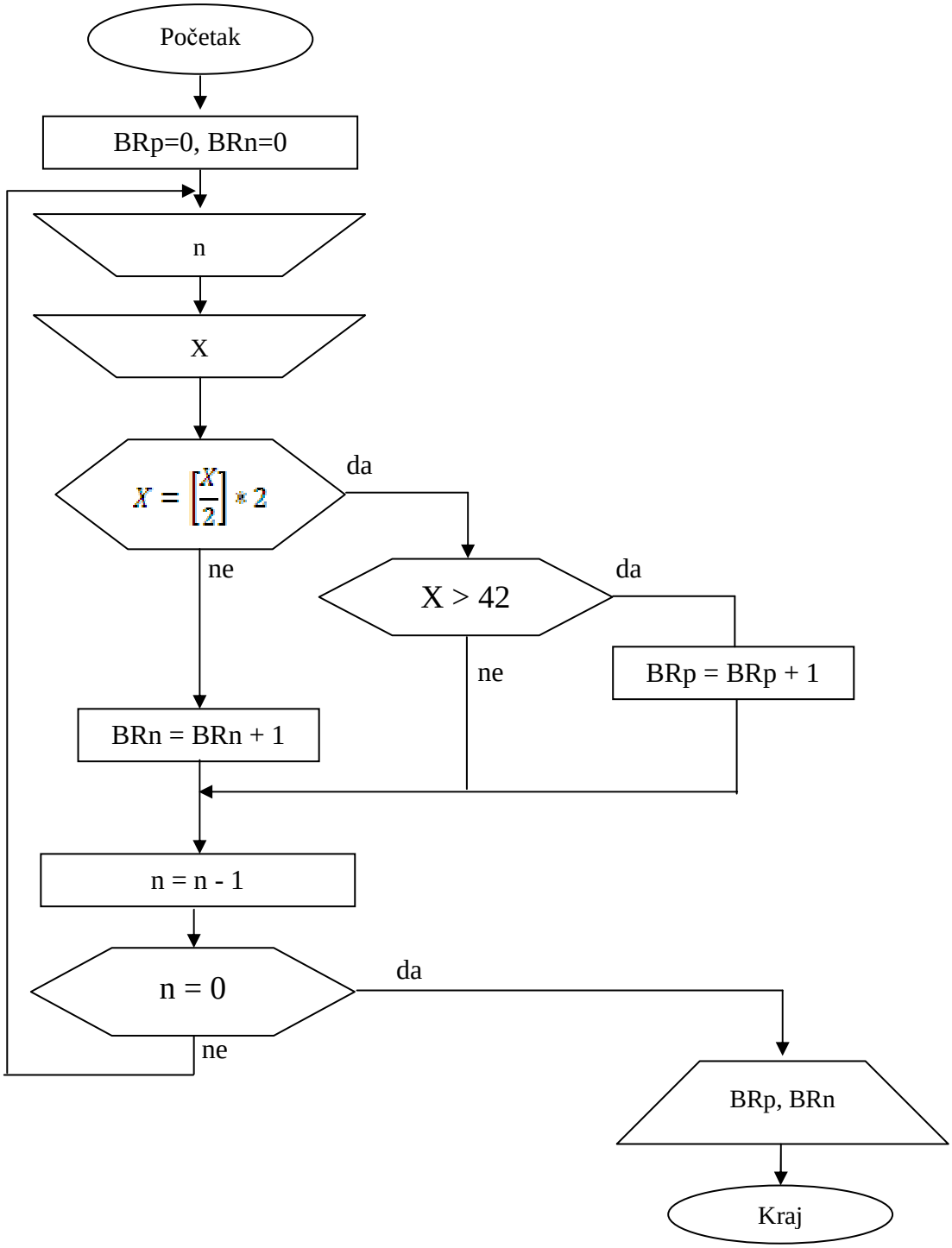
24. Unose se prirodni brojevi dok se ne unese broj 53. Sastaviti algoritam koji broji koliko je neparnih brojeva, a računa prosjek parnih brojeva. Na kraju, ako je broj neparnih unesenih brojeva veći od broja parnih unesenih brojeva, prosjek povećava za iznos razlike neparnih i parnih brojeva.



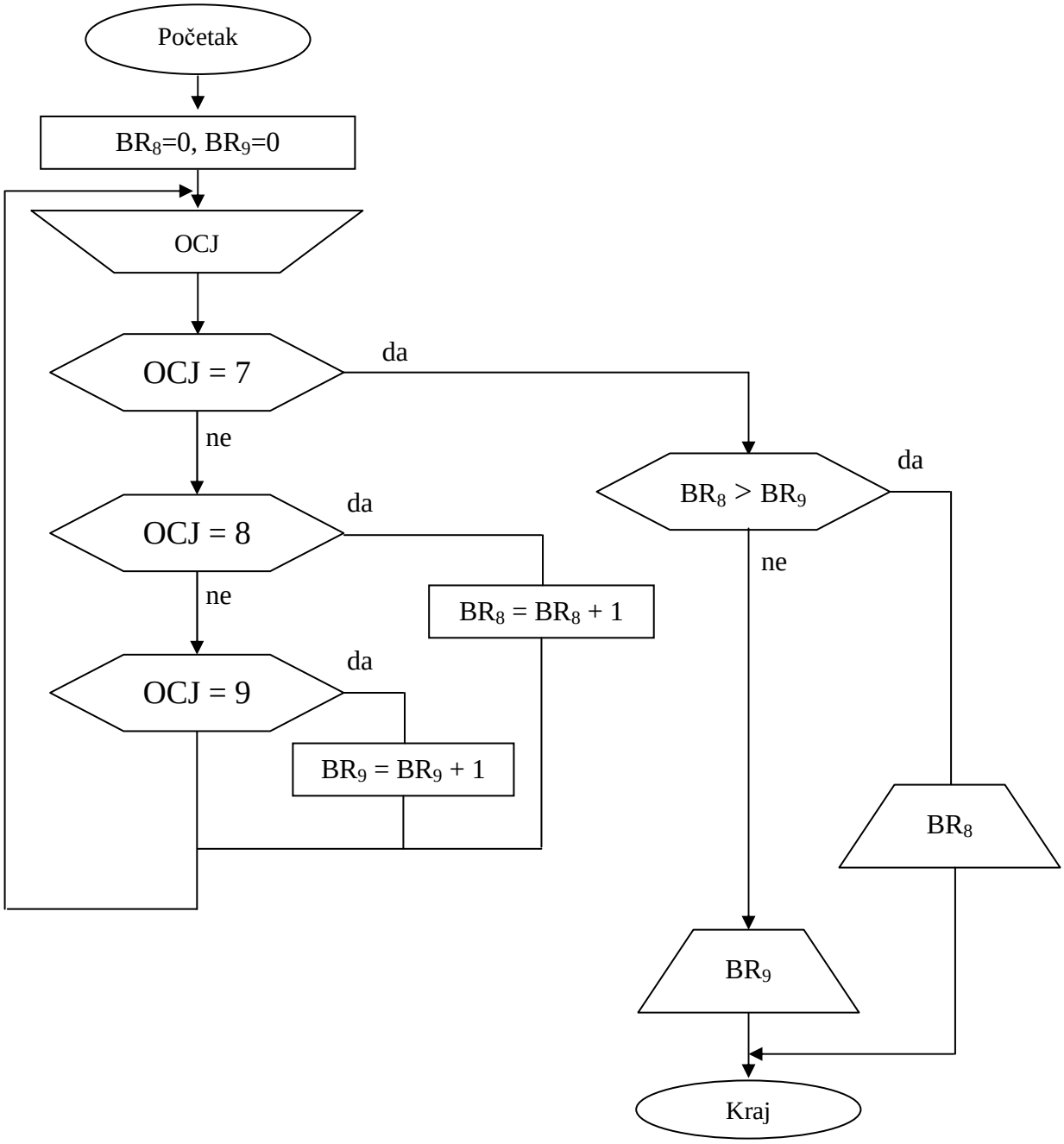
25. U banku X se unosi se n>0 štednih uloga. Sastaviti algoritam koji računa sumu štednih uloga nakon povećanja od 12% ako su veći od 580, i broji koliko je štednih uloga koji su manji od 300.



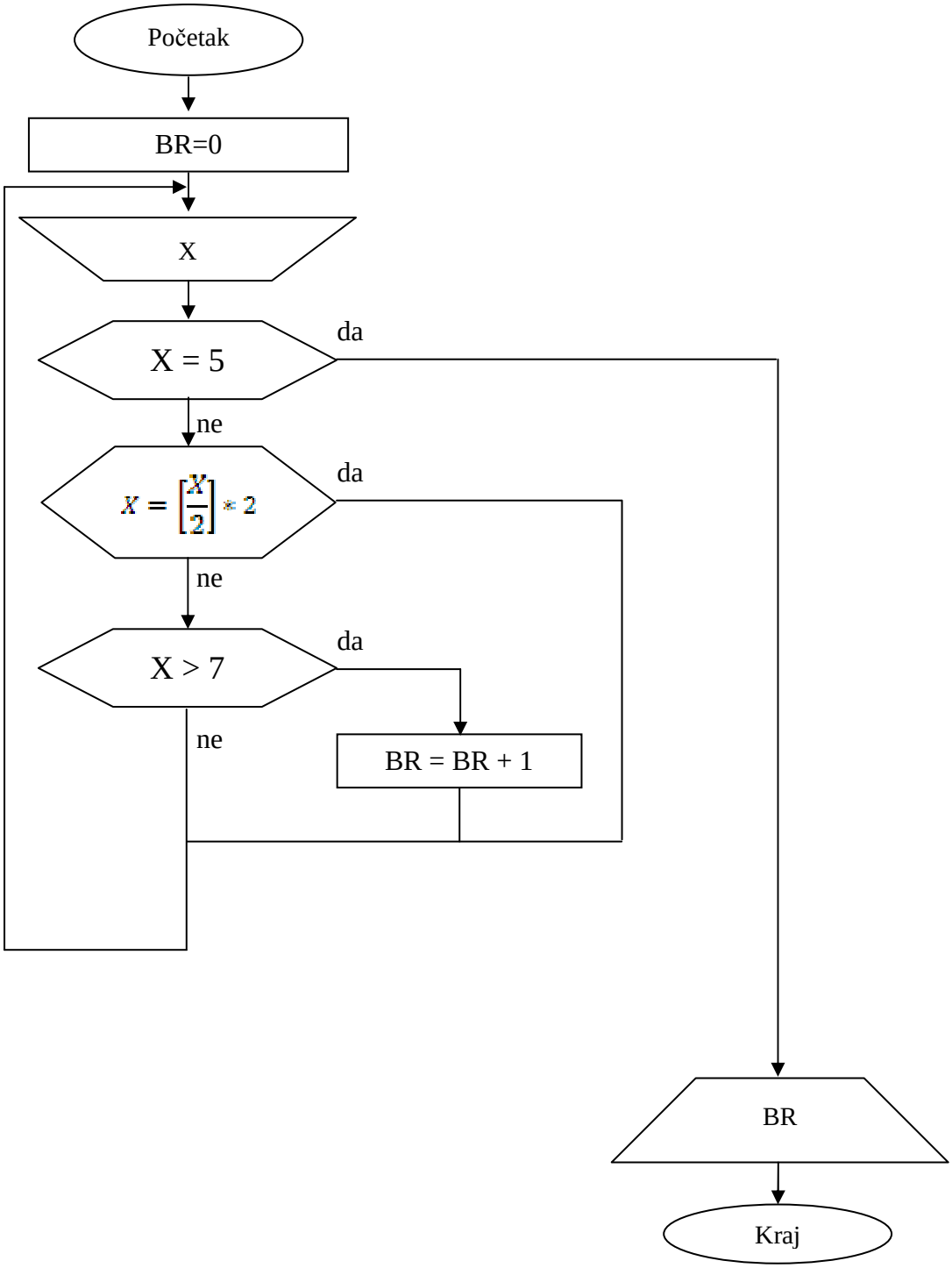
26. Unosi se n>0 prirodnih brojeva. Sastaviti algoritam koji broji parne brojeve veće od 42 i sve unesene neparne brojeve.



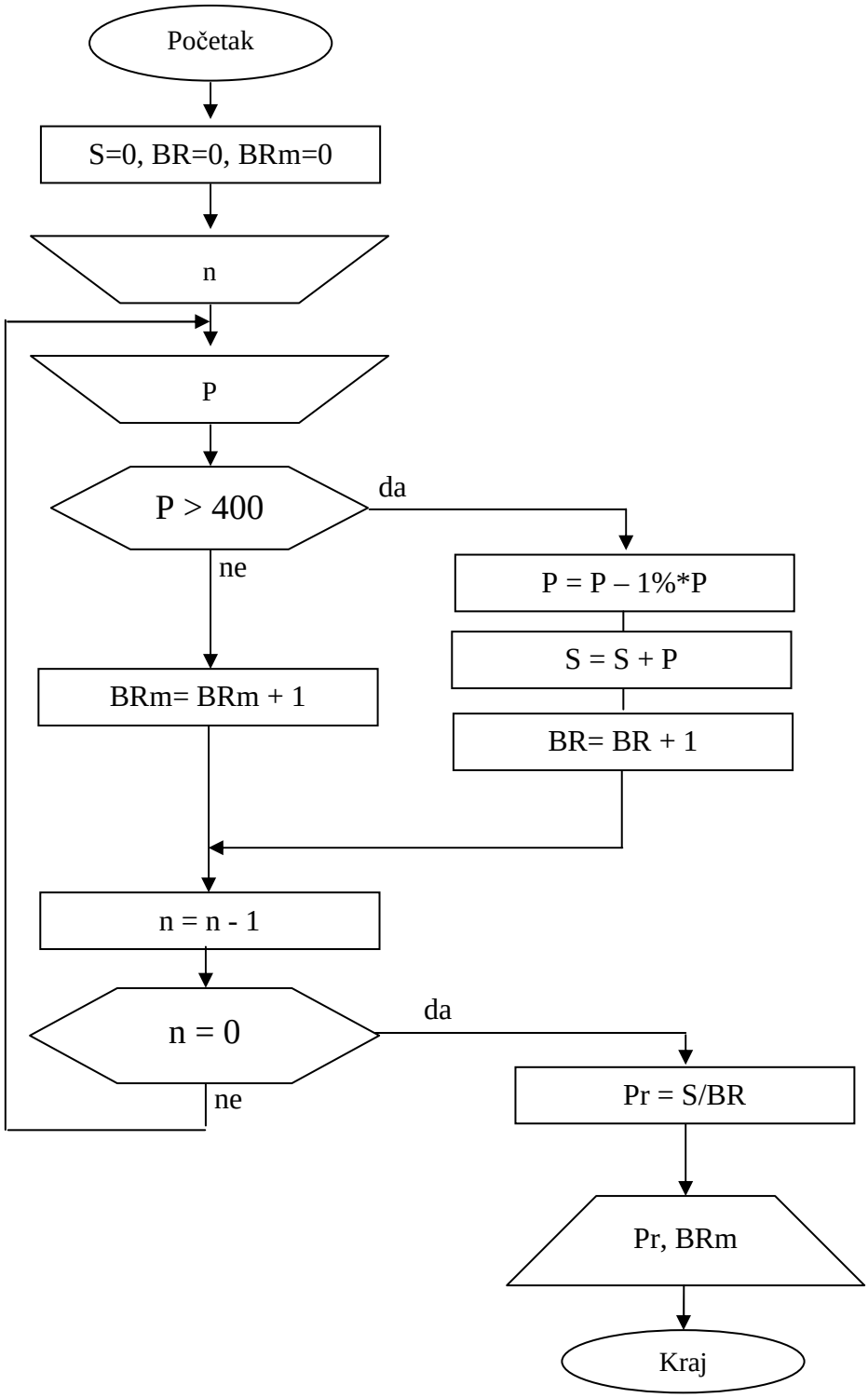
27. Unose se ocjene iz informatike dok se ne unese ocjena 7. Izbrojati koliko je unešeno ocjena 8 i 9 i prikazati broj onih ocjena kojih je više unešeno.



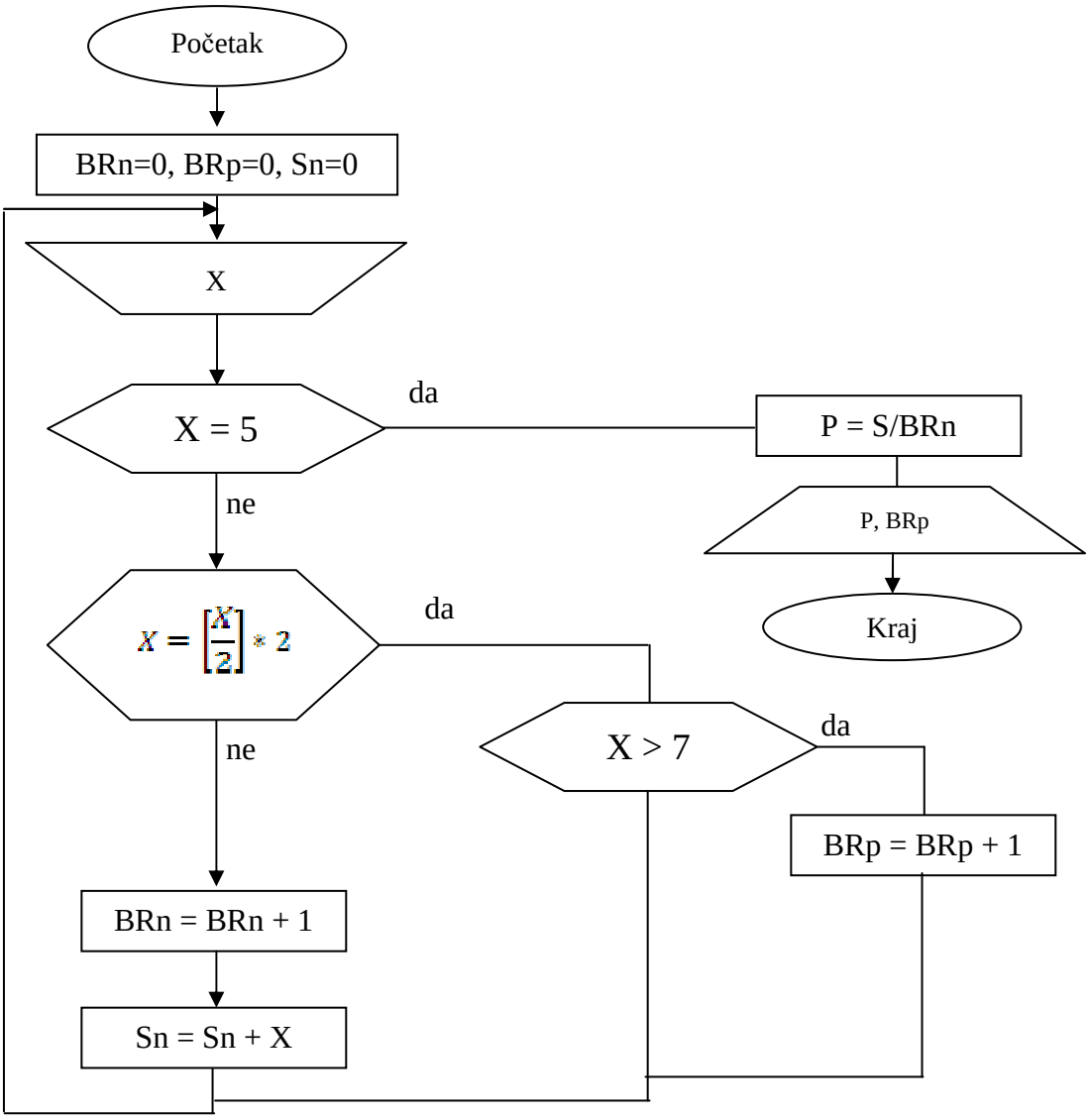
28. U bazu podataka Ekonomskog fakulteta unose se ocene iz Informatike, dok se ne unese ocena 5. Sastaviti algoritam koji za neparne unešene ocene broji koliko je onih koji su veći od 7 i prikazuje taj rezultat.



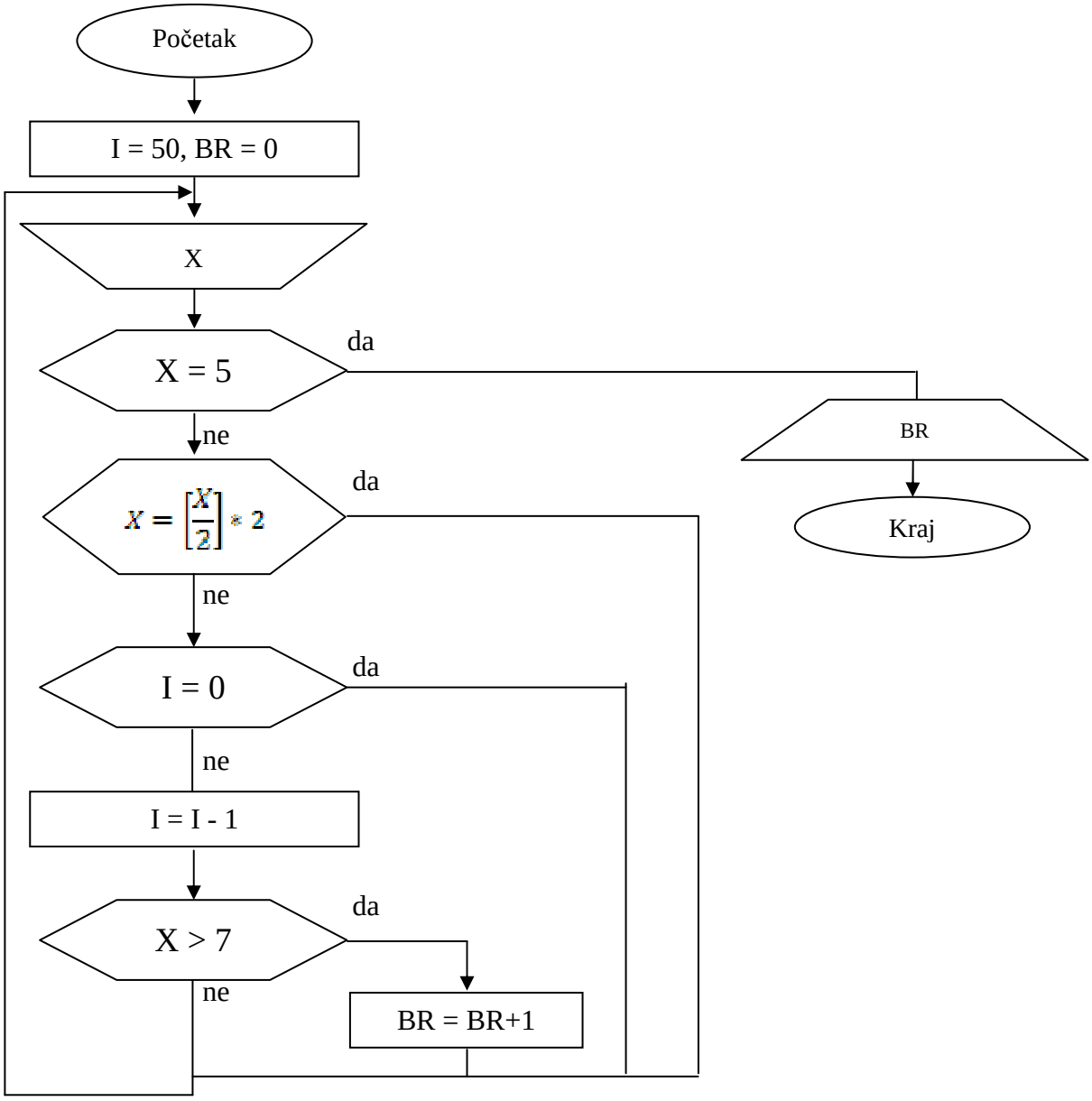
29. U bazu podataka službe računovodstva preduzeća Z unose se plate za N>0 zaposlenih. Za zaposlene čija je plata veća od 400 eura uzima se za sindikalni fond 1%. Sastaviti algoritam koji računa prosjek svih plata (nakon umanjenja) za zaposlene kojima se plata smanjuje na ime sindikalnog fonda i prikazuje broj onih kojima je plata manja od 400 eura.



30. U bazu podataka Ekonomskog fakulteta unose se ocene iz Informatike, dok se ne unese ocena 5. Sastaviti algoritam koji za neparne unešene ocene računa prosek, a za parne broji koliko je onih koji su veći od 7



31. U bazu podataka Ekonomskog fakulteta unose se ocjene iz Informatike, dok se ne unese ocjena 5. Sastaviti algoritam koji od 50 neparnih unešenih ocjena broji koliko je onih koje su veće od 7.



ZADACI ZA SAMOSTALNI RAD

1. Unose se brojevi dok se ne unese 483. Sastaviti algoritam koji broji parne brojeve manje od 824 i prikazuje taj broj i računa prosjek svih unešenih brojeva.
2. Unose se ocjene iz Matematike, dok se ne unese ocjena 5. Sastaviti algoritam koji broji koliko je unešeno ocjena, računa prosječnu ocjenu i broji koliko je studenata dobilo ocjenu 9. Na kraju prikazuje broj studenata koji su položili ispit sa ocjenom 9 kao i prosjek koji je izačunao.
3. Unose se ocjene iz računovodstva za studente koji su polagali ispit. Za studente koji su položili unose se odgovarajuće ocjene (6, 7, 8, 9 i 10) a za studente koji nisu položili unosi se ocjena 5. Sastaviti algoritam koji broji koliko je studenata položilo ispit, a koliko nije i koji računa prosječnu ocjenu studenata koji su položili ispit.
4. Unosi se n>0 podataka o izdavačima i cijenama knjiga u knjižari. Sastaviti algoritam koji cijenu knjiga izdavača Laguna umanjuje za 15% i broji koliko je knjiga ovog izdavača. Takođe, cijene knjiga izdavača Dereta povećava za 3%. Na kraju računa kolika je prosječna cijena svih knjiga, nakon svih umanjenja i povećanja cijena.
5. Unose se podaci o nazivu i cijenama svih artikala u prodavnici. Sastaviti algoritam koji traži artikal koji ima najveću cijenu i na kraju prikazuje cijenu i naziv tog artikla.
6. Sastaviti algoritamsku šemu koja računa Y prema sljedećoj formuli:

$$Y = \begin{cases} X1 + X2, & X1<X2 \\ X1 * X2, & X1=X2 \\ X1 - X2, & X1>X2 \end{cases}$$

Koja je ovo vrsta algoritamske šeme i zašto?

7. Unose se cijene pretplate, broj potrošenih impulsa kao i cijena impulsa za korisnike fixne telefonije. Ako je potrošeno više od 358 impulsa, cijena se umanjuje za 10%. Sastaviti algoritam koji broji koliko je korisnika kojima je umanjena cijena, kao i koliki je prosječan iznos telefonskog računa (napomena: iznos računa = pretplata + broj impulsa * cijena impulsa)
8. Unose se podaci o broju poslatih SMS poruka i cijeni poruke za n>0 korisnika mobilne telefonije. Cijena poruke je manja za 15% ako je korisnik poslao više od 150 poruka. Sastaviti algoritam koji računa ukupan broj poslatih poruka (za sve korisnike) kao i koliki je ukupan račun uključujući i račune kojima je cijena poruke niža.
9. Unose se plate za n>0 zaposlenih u jednom preduzeću. Sastaviti algoritam koji plate koje su manje od 200€ uvećava za 15% i broji koliko je tako uvećanih plata, i računa prosječnu platu zaposlenih u tom preduzeću (uzimajući u obzir plate nakon uvećanja).
10. Unose se prirodni brojevi dok se ne unese broj 842. Sastaviti algoritam koji prvih 20 parnih brojeva umanjuje za 50 i računa prosjek tako umanjenih brojeva. Takođe računa prosjek unešenih svih brojeva (bez umanjenja) i na kraju prikazuje oba rezultata.
11. Unosi se n>0 prirodnih brojeva. Sastaviti algoritam koji ispituje parnost/neparnost unešenih brojeva i na kraju prikazuje broj onih kojih je unešeno više (broj parnih ili neparnih).
12. Unose se ocjene iz Informatike za n>0 studenata 2. godine. Sastaviti algoritam koji broji koliko studenata će moći da polaže Informacione sisteme a koliko studenata neće moći, tj. koliko nije steklo uslov (student nije stekao uslov ako mu je ocjena iz informatike F).
13. Unose se rezultati (bodovi) za praktični dio ispita iz Informatičke ekonomije za n>0 studenata. Sastaviti algoritam koji broji koliko studenata će biti oslobođeno praktičnog dijela ispita iz Informatike (uslov za oslobađanje je da je student iz Informatičke ekonomije na praktičnom dijelu osvojio više od 25 poena). Takođe, računa prosječan broj bodova za sve studente iz Informatičke ekonomije.

14. Unose se podaci o prosječnim ocjenama po godinama studiranja i godini studija za n>0 studenata. Studenti kojima je prosječna ocjena na 3 ili 4 godini studija veća od 9 osvajaju nagradu. Sastaviti algoritam koji broji koliko studenata će osvojiti nagradu i računa prosječnu ocjenu svih studenata fakulteta na svim godinama studija.
15. Unose se podaci o destinacijama i broju milja koje putnici skupljaju za n>0 putnika. Ako je destinacija putovanja Rim, broj milja se putniku povećava za 1000, a ako je destinacija Pariz, broj milja se putniku povećava za 1500. Sastaviti algoritam koji broji koliko putnika putuje u Rim a koliko u Pariz, upoređuje ta dva broja i na kraju prikazuje veći broj. Takođe, računa koliki je ukupan broj milja sakupljen od strane svih putnika na svim letovima (uključujući i brojeve nakon povećanja).
16. Unose se ocjene iz Informatike za n>0 studenata. Sastaviti algoritam koji broji kolika je prosječna ocjena svih studenata na predmetu. Takođe, broji koliko studenata je dobilo ocjenu 9 a koliko je dobilo ocjenu 10, i prikazuje veći broj. Pored toga, prikazuje i prosječnu ocjenu.