Osnovi računarstva II

Algoritamski koraci - ciklus

Selekcija (nastavak)

- Selekcija može biti još složenija. Jedne naredbe se izvršavaju ako je prvi uslov zadovoljen; druge naredbe se izvršavaju ako prvi uslov nije zadovoljen, a drugi uslov jeste; treće naredbe se izvršavaju ako prva dva uslova nijesu zadovoljena, a treći uslov jeste. Ako nijedan od pobrojanih uslova nije zadovoljen izvršava se INAČE dio.
- Da bi ovo ilustrovali, posmatrajmo slučaj određivanja korijena kvadratne jednačine. Neka je kvadratna jednačina koja se rješava **$AX^2+BX+C=0$** i neka su zadati koeficijenti **A**, **B** i **C**.

Selekcija - Primjer

- Rješenja ove jednačine zavise od diskriminante **B^2-4AC**. Ako je diskriminanta veća od nule i **A≠0** kvadratna jednačina ima dva rješenja:

$$X_1 = \frac{-B + \sqrt{B^2 - 4AC}}{2A} \qquad X_2 = \frac{-B - \sqrt{B^2 - 4AC}}{2A}$$

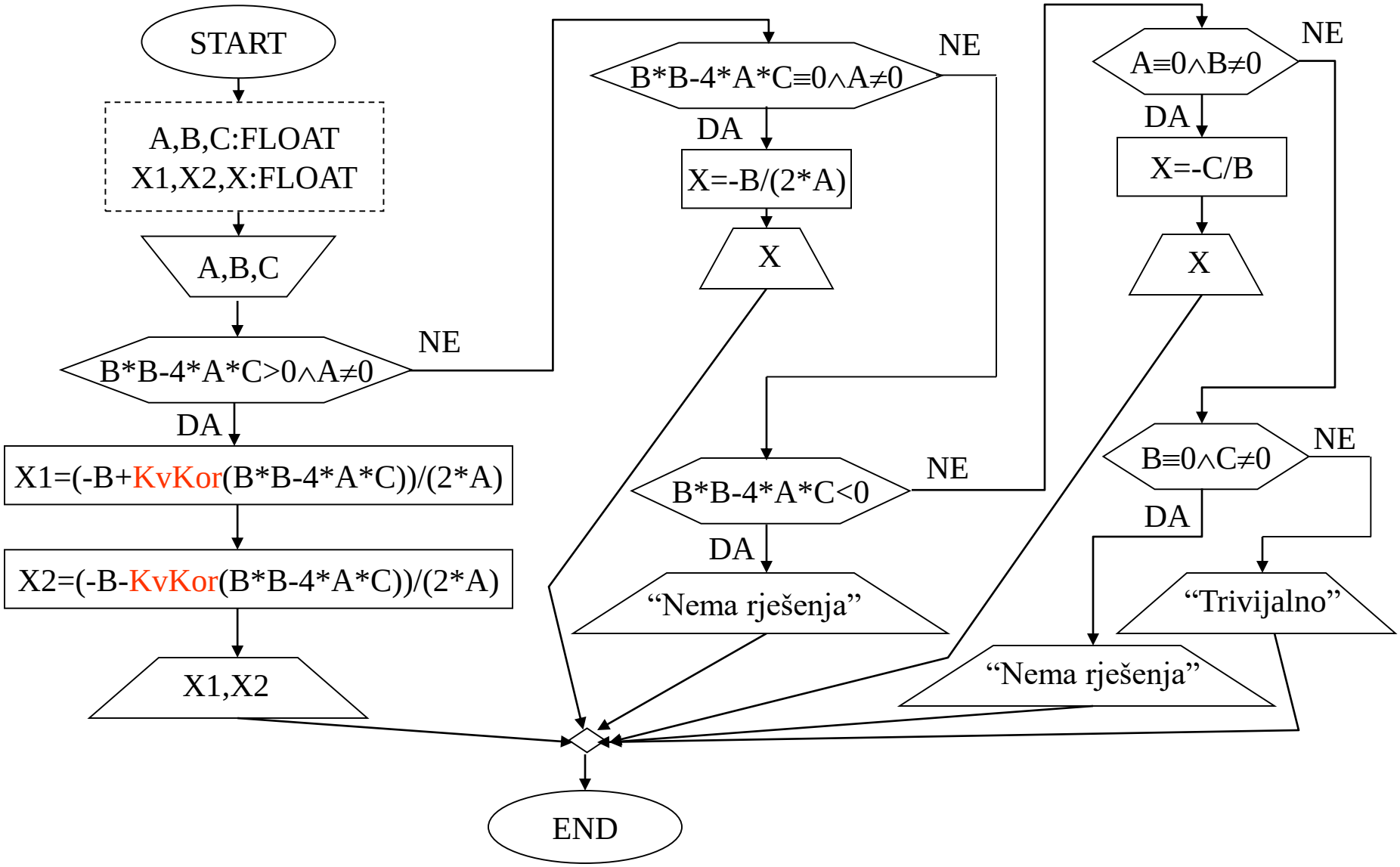
- Ako je diskriminanta jednaka nuli, **B^2-4AC≡0**, i **A≠0**, onda postoji jedno rješenje:

X=-B/(2A)

- Ako je diskriminanta manja od nule, a **A≠0**, jednačina nema rješenja u skupu realnih brojeva.

Selekcija - Primjer

- Ako je **A≡0** i **B≠0** postoji jedno rješenje **X=-C/B**.
- Ako je **A≡0** i **B≡0** i **C≠0** nema rješenja.
- Ako je **A≡0** i **B≡0** i **C≡0** rješenje je trivijalno, odnosno, svako **X** je moguće rješenje.



Pretpostavljamo da je KvKor funkcija koja daje kvadratni korjen.

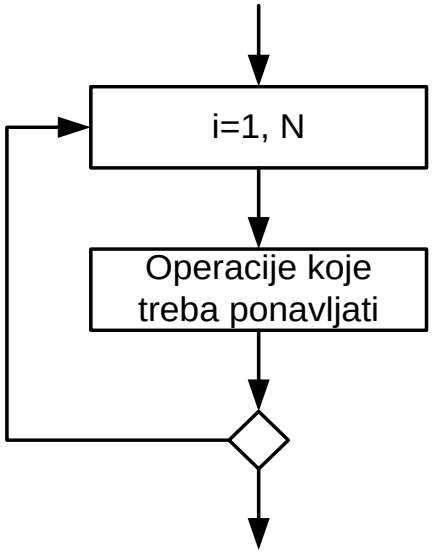
Sami napišite pseudokod za ovaj algoritam.

Ciklus

- Određeni broj algoritamskih koraka koji se ponavlja više puta se naziva ciklusom.
- Postoji više tipova ciklusa:
  - ciklusi koji se ponavljaju tačno određen broj puta,
  - ciklusi koji se ponavljaju dok je ispunjen određeni logički uslov. U zavisnosti od mjesta u ciklusu gdje se logički uslov provjerava imamo:
    - cikluse sa izlazom na početku,
    - cikluse sa izlazom na kraju,
    - cikluse sa izlazom u sredini.
- Svi ciklusi se mogu svesti na ciklus sa izlaskom na početku!

Ciklus sa unaprijed zadatim brojem izvršavanja

- Određenu operaciju treba ponoviti *N* puta gdje je *N* unaprijed zadat broj
- Koristi se brojačka varijabla
- Jedan prolazak kroz ciklus je jedna **iteracija**
- Najčešće korišćeni tip ciklusa
- Primjer:
  - Računanje zbira *N* prirodnih brojeva

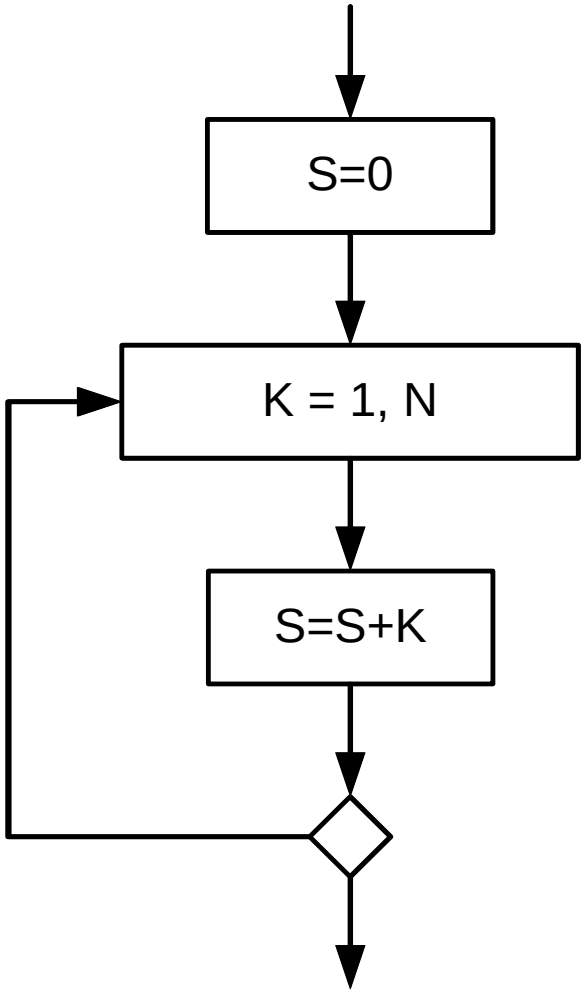



```
FOR I = 1, N
  ...
NEXT
```

Primjer:

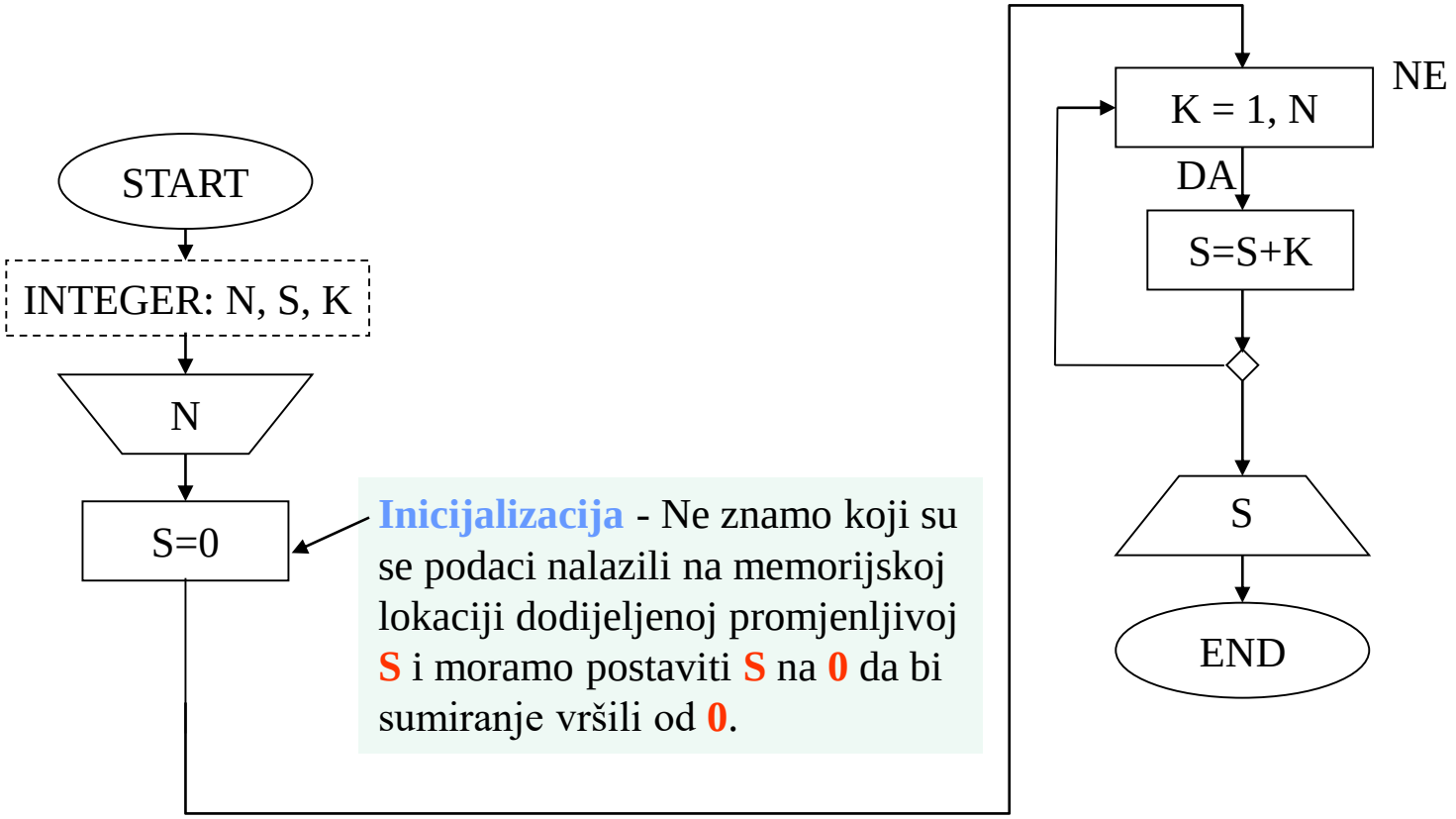
Računanje zbira prvih N prirodnih brojeva

- Rezultat je u varijabli *S*
- Inicijalizacija sume (S=0)
- Podrazumijeva se da je *N* poznat prije izvršavanja navedenih koraka i da je suma *S* iskorišćena u nastavku algoritma.



Ciklus - Primjer

- **Računanje zbira prvih N prirodnih brojeva.** Kompletan algoritam

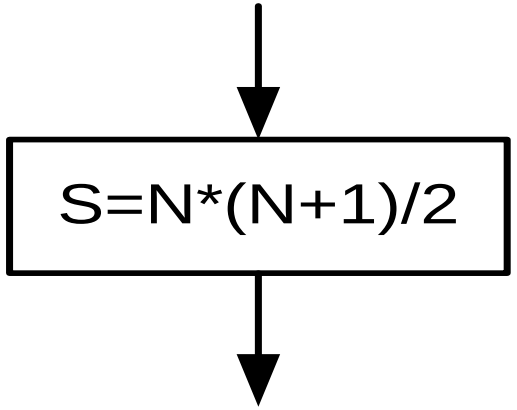


Inicijalizacija - Ne znamo koji su se podaci nalazili na memorijskoj lokaciji dodijeljenoj promjenljivoj **S** i moramo postaviti **S** na **0** da bi sumiranje vršili od **0**.

Primjer:

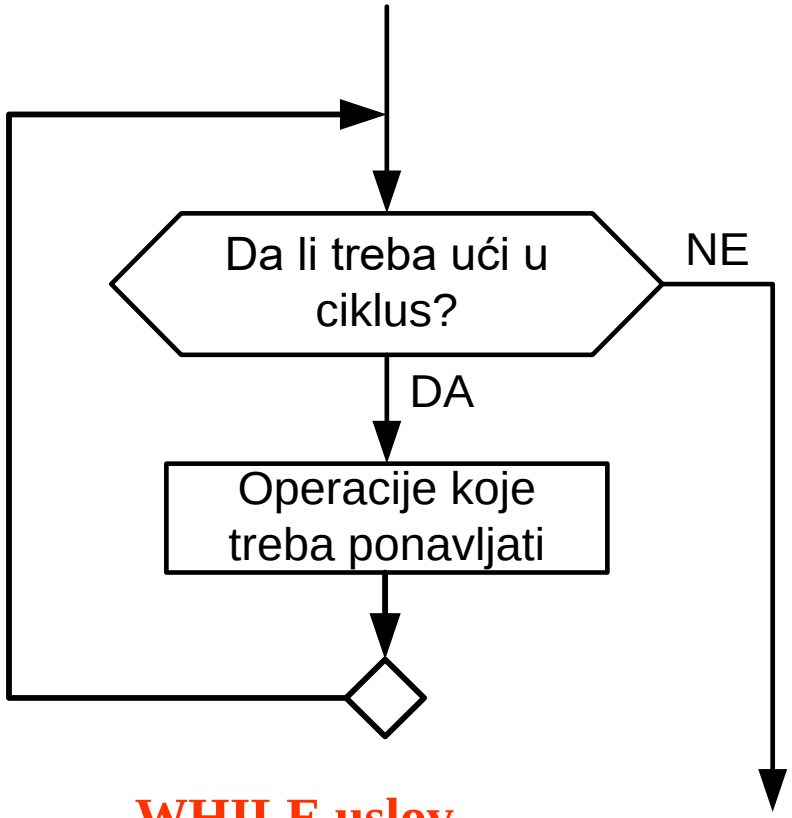
Računanje zbira prvih N prirodnih brojeva

- Alternativno rješenje bez upotrebe ciklusa
- Smanjeni broj računskih operacija
- Zahtijeva detaljniju analizu posmatranog problema



Ciklus sa izlazom na početku

- Broj ponavljanja ciklusa može biti 0 (na samom početku uslov nije zadovoljen)
- Primjena:
  - Polazimo od mogućeg rješenja problema i popravljamo ga u svakoj iteraciji (ciklusu) sve dok ne budemo zadovoljni



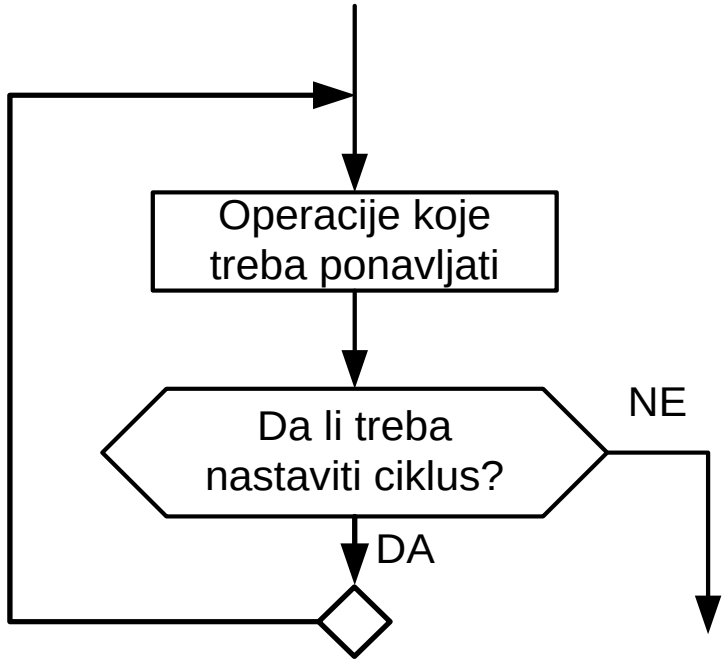
WHILE uslov

...

ENDWHILE

Ciklus sa izlazom na kraju

- Broj ponavljanja ciklusa je najmanje 1
- Primjena:
  - Unos podataka koji moraju zadovoljiti postavljene kriterijume
  - Primjer: od korisnika tražimo unos pozitivnog cijelog broja



DO

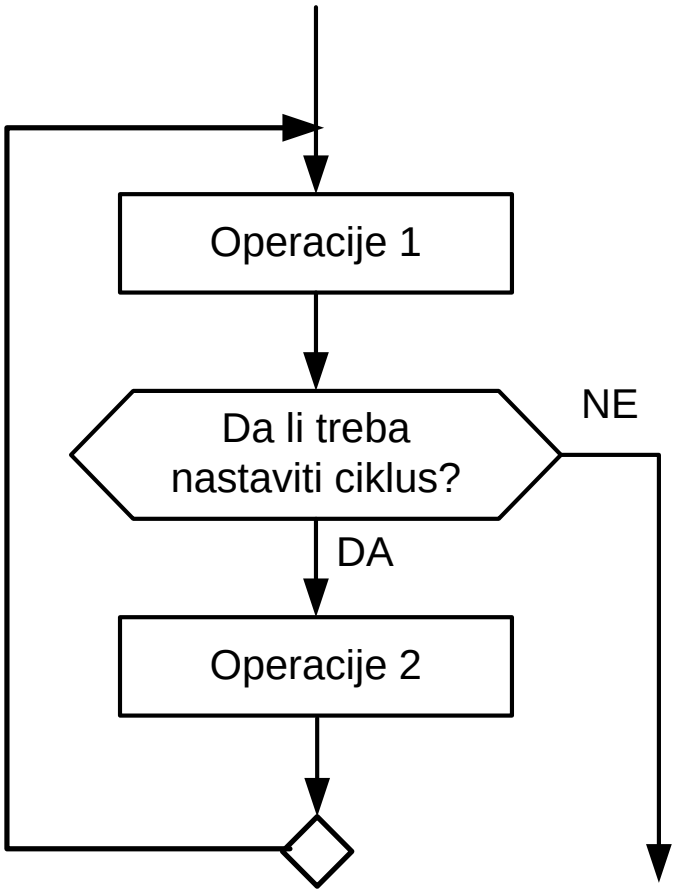
...

WHILE uslov

Ciklus sa izlazom u sredini

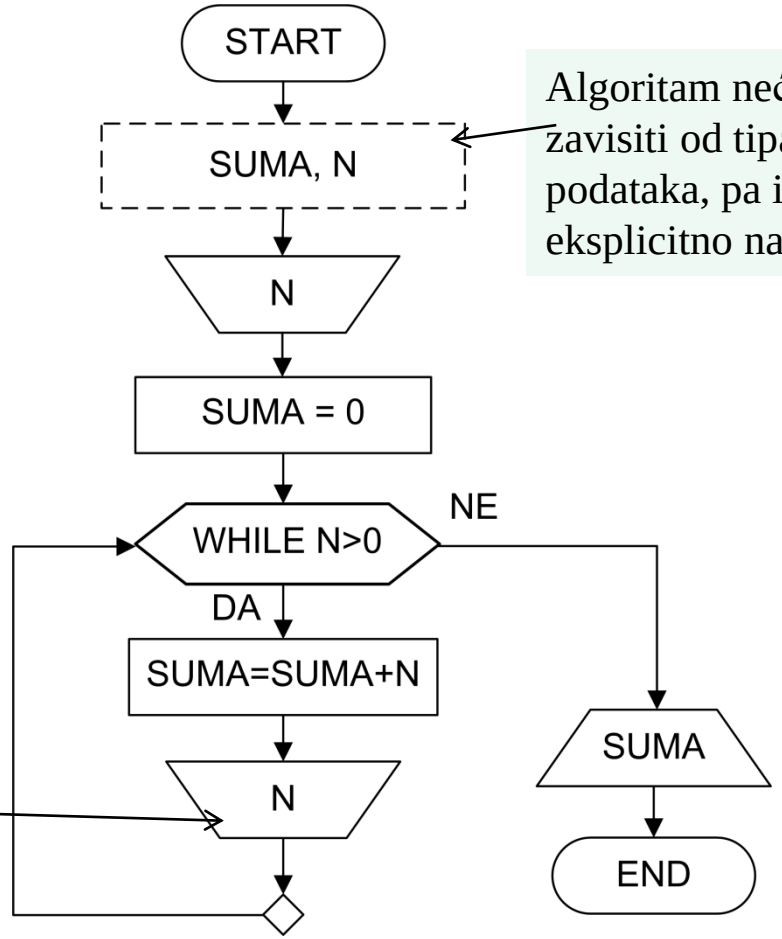
- Dio ciklusa mora biti izvršen, a nakon toga se donosi odluka da li nastaviti sa izvršavanjem drugog dijela.
- Svaki ciklus se može svesti na ciklus sa izlazom na početku

```
DO
  ...
  IF uslov THEN EXITDO
  ...
LOOP
```



Ciklus – Primjer 2

- Prethodni primjer je primjer ciklusa koji se izvršava tačno određen broj puta.
- Ovaj primjer je ciklus koji se izvršava dok je zadovoljen uslov.
- **Izvršiti sumiranje brojeva koje korisnik zadaje sve do unosa negativnog broja. Negativan broj ne sabirati.**

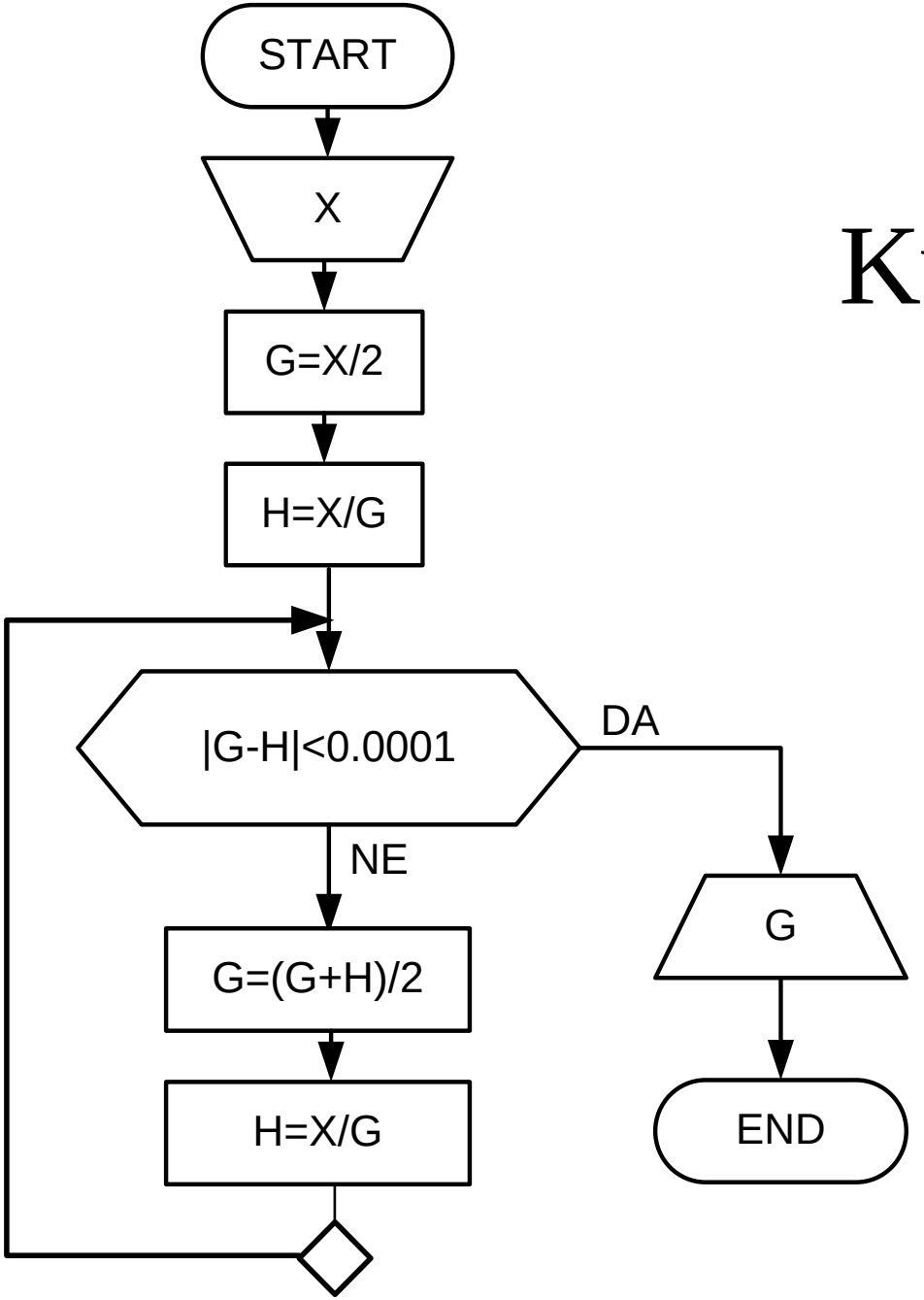


Algoritam neće zavisiti od tipa podataka, pa isti nije eksplicitno naveden.

U petlji se mogu i unositi i ispisivati podaci

Ciklus – Primjer 3

- Kao primjer ciklusa (onog koji se izvršava nepoznat broj puta, odnosno, dok je zadovoljen određeni logički uslov) navodimo Njutnov algoritam za približno određivanje kvadratnog korijena realnog broja.
- **Korak 1.** Neka tražimo kvadratni korijen broja X. Kao početno pogađanje njegovog kvadratnog korijena uzmimo G=X/2. Izračunajmo broj H kao H=X/G.
- **Korak 2.** Ako su H i G veoma slični to znači da je H*G približno jednako X, odnosno da su G i H približno jednaki kvadratnom korijenu broja X.
- **Korak 3.** Ako nijesu približno jednaki, onda treba uzeti da je novo G jednako G=(G+H)/2 i ponovo sračunati H kao H=X/G.
- **Koraci 2 i 3** se ponavljaju dok se ne dođe do rješenja.



Ciklus
Kvadratni korijen

X,G,H realni brojevi

X,G,H: FLOAT

Za *X=9* biće *G=4.5, H=2*

Iteracije:

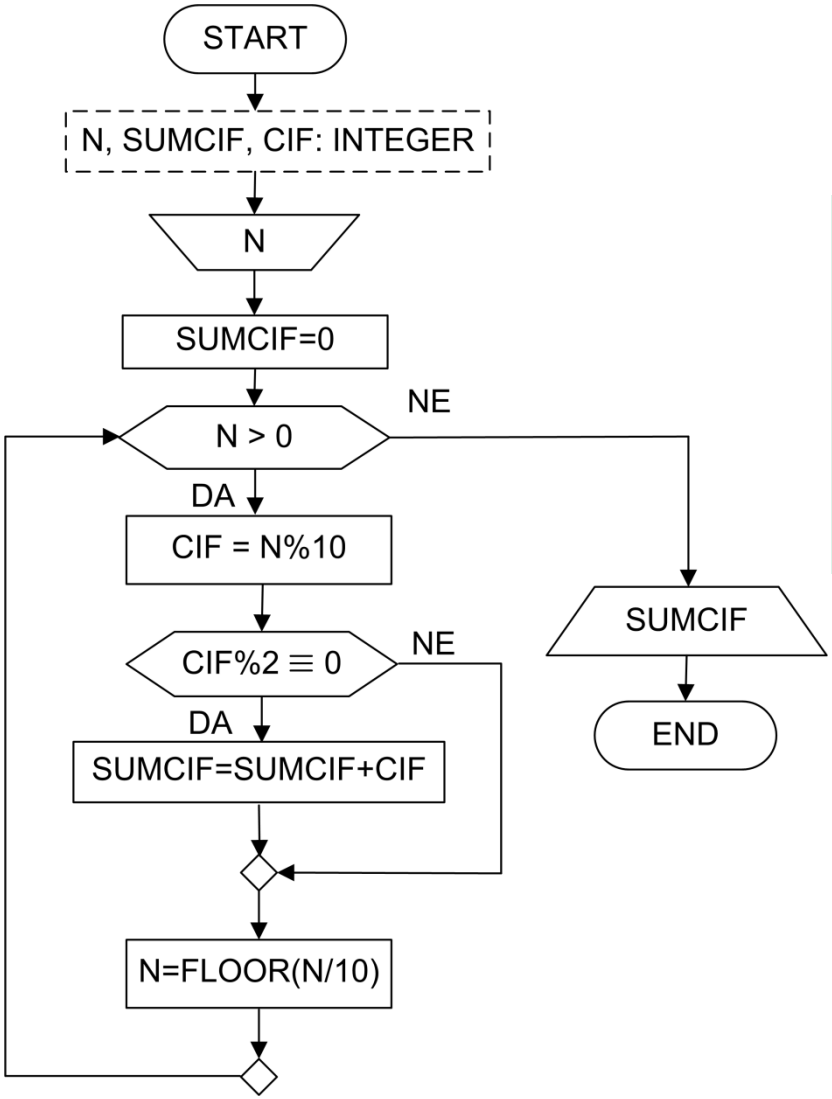
1. *G=3.25, H=2.77*
2. *G=3.01, H=2.99*
3. *G=3, H=3*

Ciklusi i selekcija

- Unutar jednog ciklusa se može nalaziti proizvoljna sekvenca naredbi.
- Može se dokazati da se svaki algoritamski rješiv problem može riješiti korišćenjem samo **sekvenci, selekcija i ciklusa**.
- Nekoliko ciklusa mogu jedan za drugim da čine sekvencu.
- Unutar ciklusa se može nalaziti selekcija.
- Unutar selekcije se može nalaziti ciklus.
- Unutar selekcije se može nalaziti druga selekcija
- Unutar ciklusa se može nalaziti drugi ciklus (to se naziva ugnježdenje ciklusa).
  - Važno pravilo je da se prvo završava unutrašnji dio (unutrašnji ciklus ili selekcija), pa tek onda spoljašnji, tj. ciklusi i selekcije se ne mogu sjeći.

Ciklus - primjer 4

- Unosi se prirodan broj N. Odrediti sumu njegovih parnih cifara.



Unosi se **N=249**

Prvi prolaz:	SUMCIF = 0,	N = 24
Drugi prolaz:	SUMCIF = 4,	N = 2
Treći prolaz:	SUMCIF = 4+2,	N = 0

% računa ostatak pri dijeljenju

FLOOR zaokružuje na najbliži manji cijeli broj.

Nizovi i matrice

- Niz je **složen tip podatka.**
- Programski jezici često rade sa većom količinom podataka istog tipa.
- Ti podaci se po potrebi mogu smjestiti u niz.
- Elementi niza cijelih brojeva dužine **N** se mogu obilježavati sa: **a[1], a[2], ..., a[N] ili a(1), a(2), ..., a(N).** Brojevi 1, 2, ..., N predstavljaju redne brojeve elemenata niza ili **indekse niza**.
- Napomenimo da različiti programski jezici usvajaju drugačije notacije za indeksiranje nizova.
- Kod matrica dimenzija **MxN** elementi su indeksirani kao:
  a[1,1], a[1,2], ..., a[1,N],
  a[2,1], a[2,2], ..., a[2,N],
  ...
  a[M,1], a[M,2], ..., a[M,N].

Nizovi deklaracija i indeksiranje

- Niz se deklariše na sljedeći način: **TIP : X[50].** U našim primjerima TIP će biti **INTEGER** ili **FLOAT.**
- Broj 50 u deklaraciji (naravno, može biti bilo koji pozitivan cijeli broj) predstavlja najveći očekivani broj podataka, bez obzira na to koliko se podataka stvarno koristi u nizu.
- Uvešćemo i dodatni podatak koji predstavlja pravu dužinu niza. Taj podatak se učitava u algoritmu.
- Postavlja se pitanje na koji način efikasno raditi sa elementima niza (učitavanje, modifikacija, štampanje) kad se dužina može mijenjati pri svakom izvršenju programa.
- Odgovor – Pomoću ciklusa!
- Uvešćemo pomoćnu promjenljivu, recimo I, i **elementima niza ćemo pristupati sa X[I], pri čemu se I mijenja od 1 do N, gdje je N prava dužina niza**.

Operacije sa elementima niza

- Sa elementima niza su dozvoljene sve operacije koje su dozvoljene u radu sa elementarnim podacima tipa kojem pripadaju elementi niza:

$$b[1] = b[2] - b[3]$$

$$a[2,3] = a[1,2] - b[1]$$

$$b[1] > 2$$

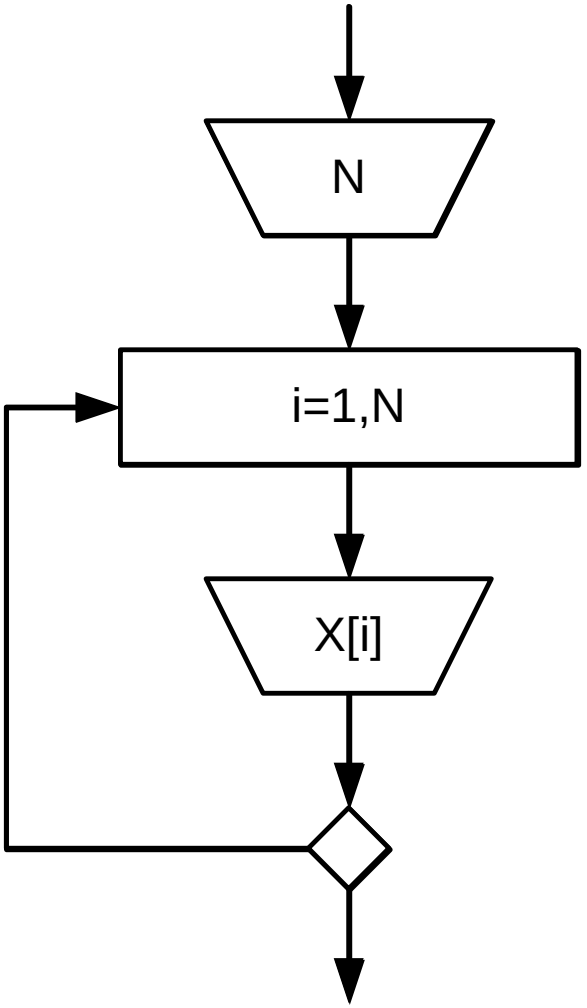
Niz karaktera

- Niz karaktera se naziva **string.**
- Sa članovima niza karaktera mogu da se vrše sve operacije koje se mogu vršiti sa podacima tipa karakter.
- Jedna bitna razlika u odnosu na nizove cijelih i realnih brojeva je ta da se podaci koji čine niz brojeva učitavaju sa tastature računara jedan po jedan, i na isti način prikazuju na ekranu, dok se niz karaktera može učitati i prikazati odjednom.

Primjer:

Unos elemenata niza X

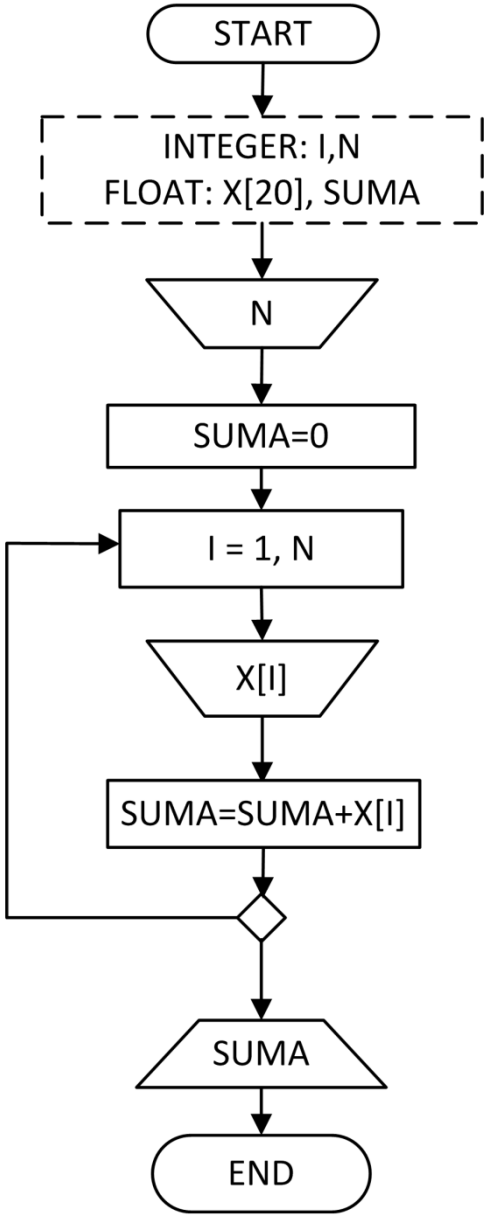
- Broj elemenata niza mora biti unaprijed poznat
- U jednom ulaznom koraku može se učitati samo jedan element niza




```
INPUT N
FOR  i = 1, N
   INPUT X[i]
NEXT
```

Nizovi - Primjer 1

- Nacrtati algoritam kojim se unosi niz od N realnih elemenata i štampa suma elemenata niza.
- Mogu se u jednoj petlji učitati elementi niza, a u drugoj sabirati, ali je optimalnije da se u istoj petlji i učitavaju i sabiraju.



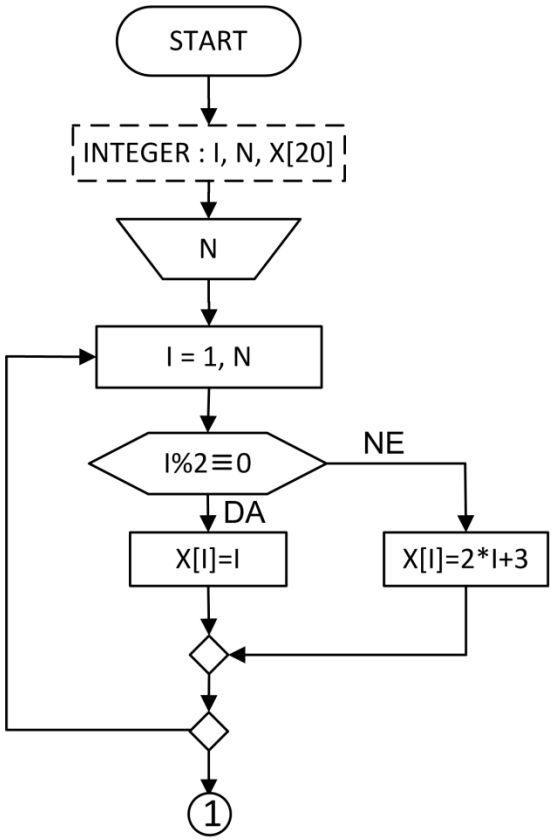
Nizovi - Primjer 2

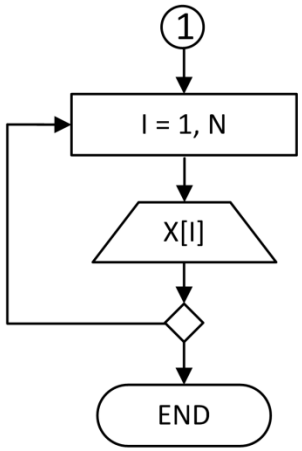
- Nacrtati algoritam kojim se formira i štampa niz X sastavljen od N elemenata datih sljedećom relacijom:

$$X[I]=\begin{cases} I, & \text{za parno I} \\ 2I+3, & \text{za neparno I.} \end{cases}$$

Na primjer, za **N=5**, niz je

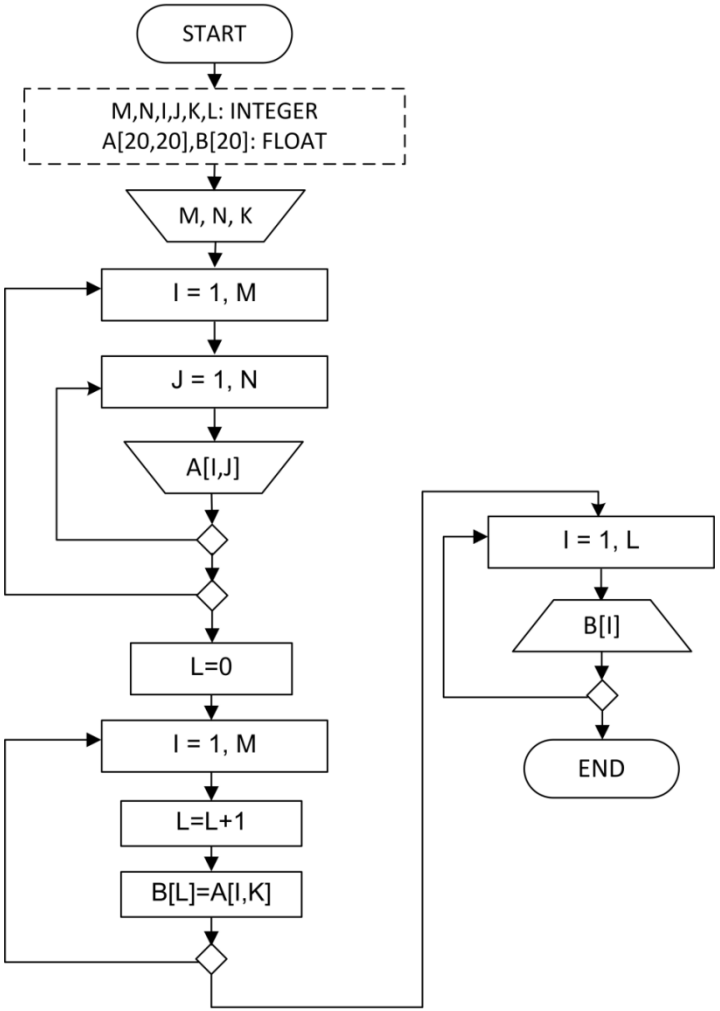
X = 5, 2, 9, 4, 13

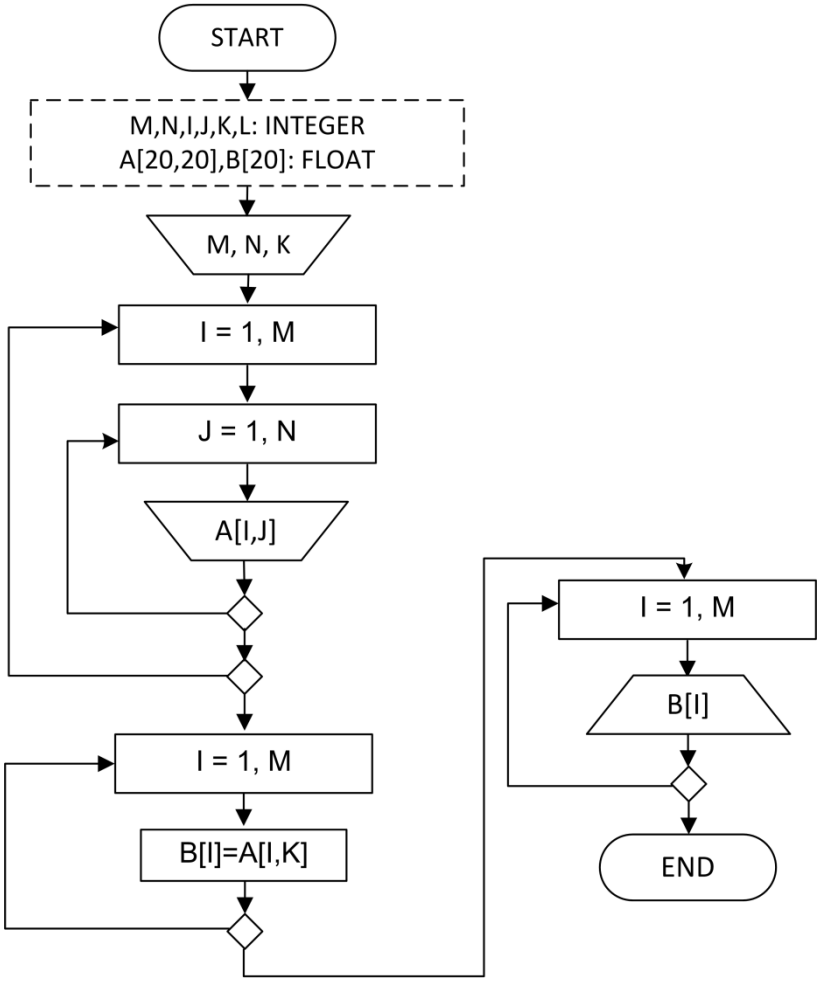




Matrice – Primjer 1

Za datu matricu A, formirati niz B sastavljan od elemenata kolone koju zadaje korisnik.





Primjeri za vježbu - Matrice

- Matrice su izuzetno pogodne za uvježbavanje do sada naučenih elemenata algoritama.
- Evo vam nekoliko zadataka za vježbanje:
  - Sabiranje, oduzimanje i množenje dvije matrice, sa provjerom da li im dimenzije omogućavaju ove operacije.
  - Odrediti sumu (ili proizvod) elemenata matrice;
  - Odrediti maksimalni (ili minimalni) element matrice;
  - Kreirati vektor koji se sastoji od zadate vrste (ili kolone) matrice;
  - Odrediti sumu (ili proizvod) svih kolona (ili vrsta) matrice.

Primjeri za vježbu - Matrice

- Kreirati vektor koji se sastoji od elemenata matrice sa glavne (ili sporedne) dijagonale;
- Provjeriti da li su učitane kvadratne matrice inverzne;
- Sračunati trag matrice (trag matrice je suma elemenata glavne dijagonale matrice);
- Provjeriti da li je matrica simetrična;
- Odrediti transponovanu matricu date matrice, i slično.