Faze programiranja

Pojam algoritma

- **Muhammed ibn Musa al Khowarizmi** Bagdadski pisac, matematičar, astronom i geograf
- 852. godine napisao knjigu **Kitab al jabr w'al-muqabala**
- Od riječi **al jabr** nastala riječ **algebra,** slično zbog pogrešnog izgovaranja riječi od njegovog imena **al Khowarizmi** nastala riječ **algoritam.**

Pojam algoritma

- **Algoritam je konačan niz preciznih koraka koji vode do rješenja nekog problema.**

- Koraci bi trebali biti toliko jednostavni da za njihovo izvršenje nije potrebna nikakva inteligencija.

- Pisanje programa izvršava se u nekoliko etapa:
  - **Postavka problema**
  - **Skica rješenja**
  - **Izrada algoritma**
  - **Prevođenje algoritma u izvorni kod odgovarajućeg programskog jezika**

Pojam algoritma

- **Koraci moraju biti jednostavni tako da ih može izvršavati mašina.**

- **Razvijati algoritamski način razmišljanja je poželjno bavili se programiranjem ili ne.**

- Algoritme obično zapisujemo na dva načina:
  - Dijagramom toka
  - Pseudo kodom

Pojam algoritma

- Svaki algoritam je sastavljen od osnovnih algoritamskih postupaka:

- **Slijed (**jedan ili više koraka koji se izvode u nizu**)**

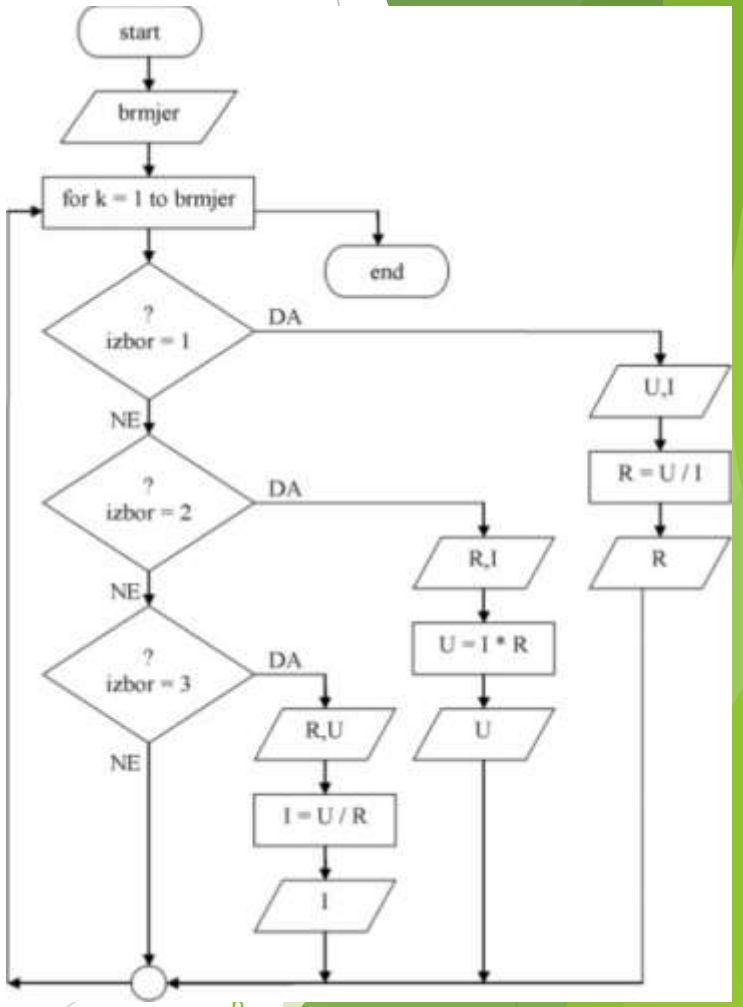
- **Grananje (**odluka koja **slijed** vodi na više strana u ovisnosti od uvjeta**)**

- **Petlje** (**slijed** algoritamskih kora koji se izvrsava određeni broj puta)

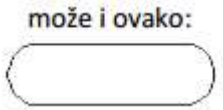
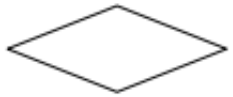
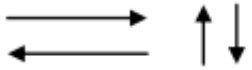
Zapisivanje algoritma,dijagram toka

- Dijagram toka je **grafički** prikaz algoritma.
- Zapisivanje se vrši **međunarodno dogovorenim simbolima**
- Prednosti ovoga načina zapisa:

➢ **Jednostavan zapis**
➢ **Preglednost**
➢ **Olakšano pronalaženje grešaka**
➢ **Jednostavna analiza i poređenje sa drugim problemima**



Zapisivanje algoritma,dijagram toka

Element	
	Početak ili kraj algoritma 
	Unos podataka, prikaz rezultata
	Može se koristiti za prikaz rezultata
	Može se koristiti za unos podataka
	Jednostavna radnja (naredba)
	Odluka
	Slijed programa
	Nastavak dijagrama (poveznica)

Zapisivanje algoritma,dijagram toka

- Dijagram toka je **grafički** prikaz algoritma.

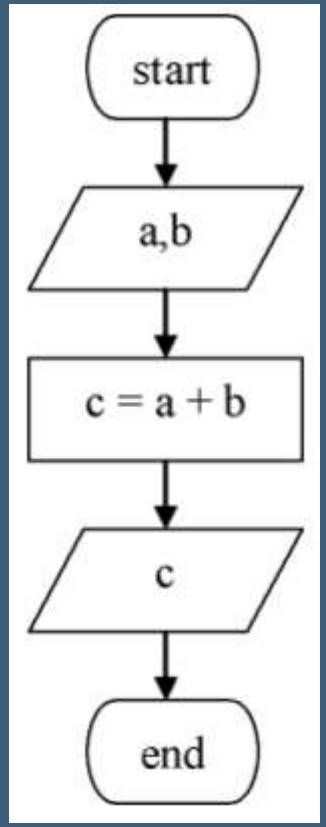
- Zapisivanje se vrši **međunarodno dogovorenim simbolima**

- Prednosti ovoga načina zapisa:

➢ **Jednostavan zapis**
➢ **Preglednost**
➢ **Olakšano pronalaženje grešaka**
➢ **Jednostavna analiza i poređenje sa drugim problemima**

Primjer 1 - Sabiranje brojeva

- Napisati dijagram toka za sabiranje brojeva. Program treba da traži od korisnika unos dva broja *(a i b)* a kao rezultat treba da ispiše njihov zbi (neko *c*).



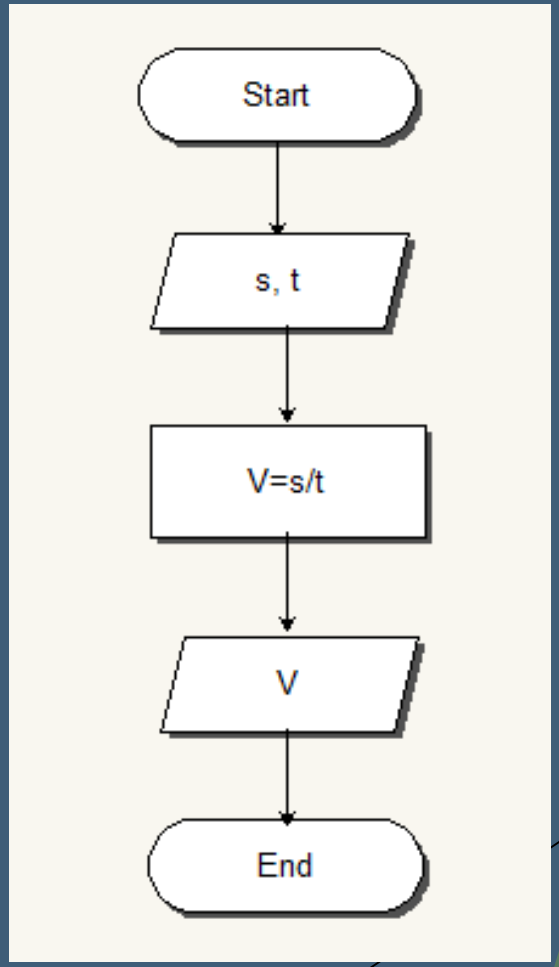
Zapisivanje algoritma,dijagram toka

- **Slijed**

Dijagram toka kakav smo razmatrali za sabiranje dva zbroja često se naziva **slijed** jer naredbe slijede pravolinijski jedna iza druge. Stariji programeri znaju reći da program "propada" do kraja.
U sljedečem primjeru imamo još jedan jednostavan slijed.

Primjer 2 – Računanje brzine

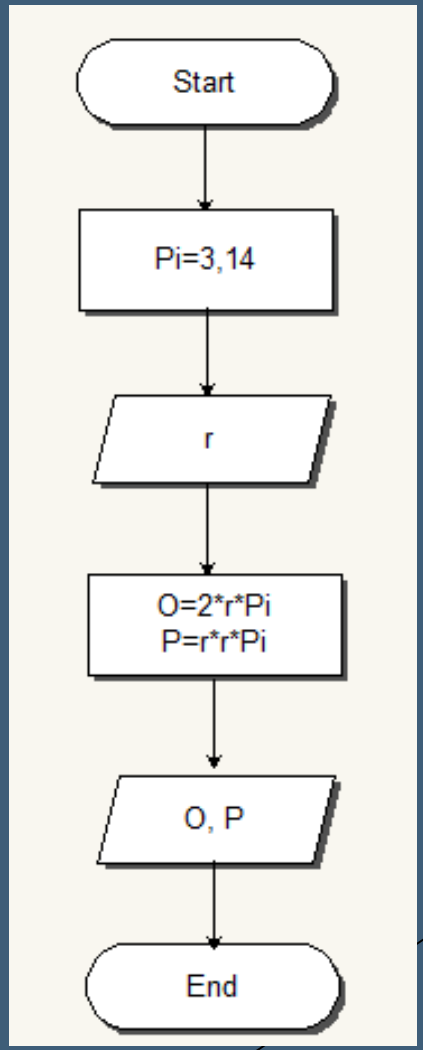
- Iz osnovne škole sjećamo se da je brzina nekog tijela zapravo prijeđeni put u jedinici vremena.
- Matematički bismo to zapisali kao **V = s / t**, gdje je **V** brzina, **s** prijeđeni put i **t** vrijeme. To je tako jednostavno da nam nije potrebna nikakva priprema i gruba skica rješenja. Možemo odmah raditi algoritam.



Primjer 3 – Računanje obima i površine kruga

- Pošto nemamo grčkih slova u opticaju umjesto π mi ćemo pisati Pi. Formule koje su nam potrebne su:
- $O = 2 \cdot r \cdot Pi$
- P=$r * r * Pi$

(r^2 ćemo pisati kao r*r)



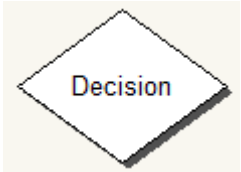
Zadaća

- Napraviti dijagram toka za problem racunanja povrsine i obima pravougaonika.

(Formule koje koristimo su $O = 2 \cdot a + 2 \cdot b$ i $P = a \cdot b$)

Grananje

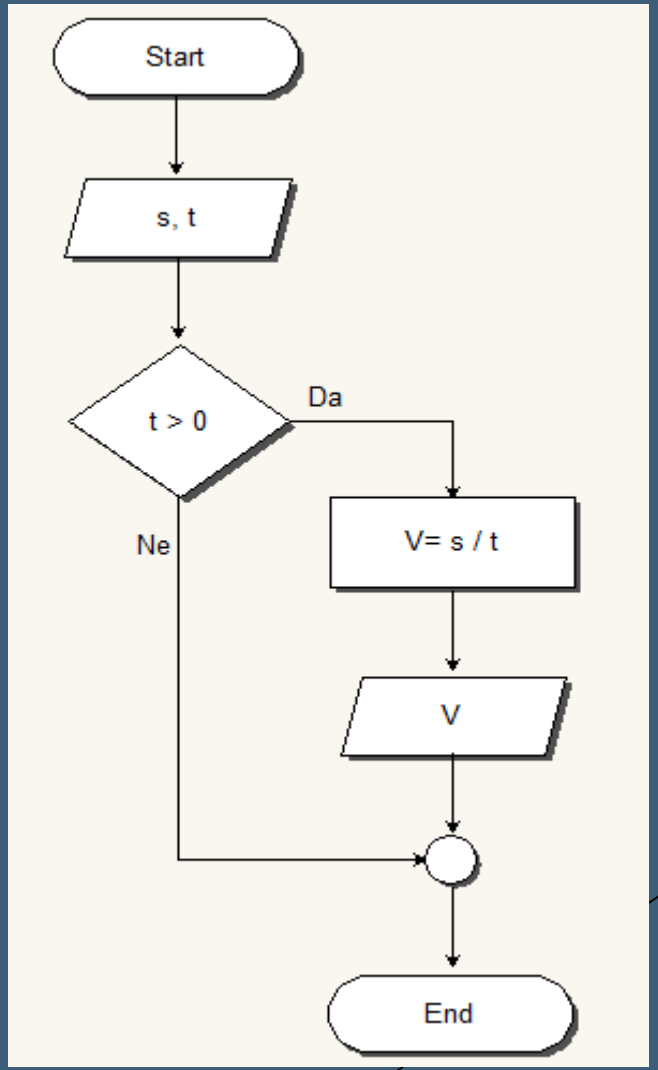
- Malo je problema u programiranju koji se mogu riješiti pravolinijskim slijedom. Ponekad su potrebna poređenja i donošenje odluka. Takva mjesta u algoritmu se nazivaju grananja.
- Simbol koji se koristi za grananje je romb



- U sredinu se postavlja **uvijet** (logički)
- Blok grananja uvijek ima **jedan ulaz** i najmanje **dva izlaza**
- Ako imamo samo dva izlaza onda logički uvjet ima samo dva odgovora **DA** i **NE**

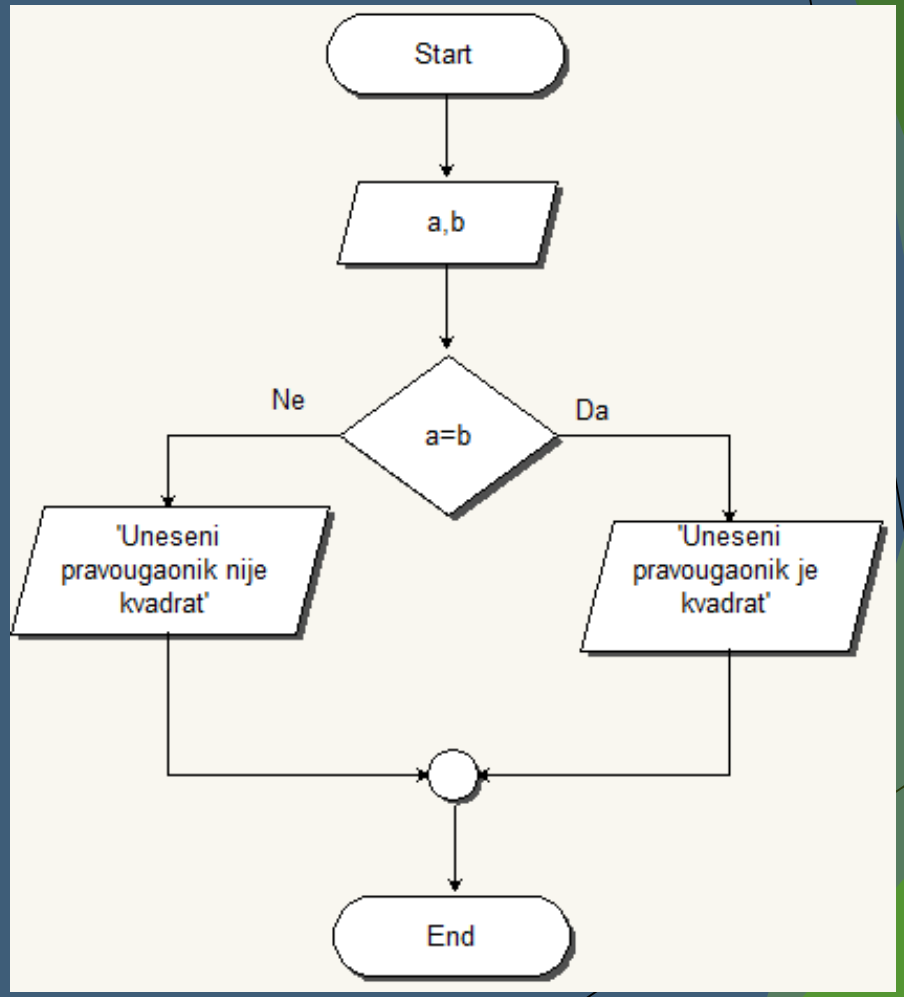
Primjer 4 – djeljenje nulom

- Vratimo se na primjer racunanja brzine. Sta ako je vrijeme t bilo 0.
- Iz matematike znamo da se sa 0 ne smije dijeliti.
- Treba prepraviti postojeci dijagram toka tako da program završi ukoliko se desi da je t=0.



Primjer 5 – pravougaonik-kvadrat

- Nacrtati dijagram toka za algoritam koji ispituje da li je neki pravougaonik kvadrat.



Primjer 6 – paran/neparan

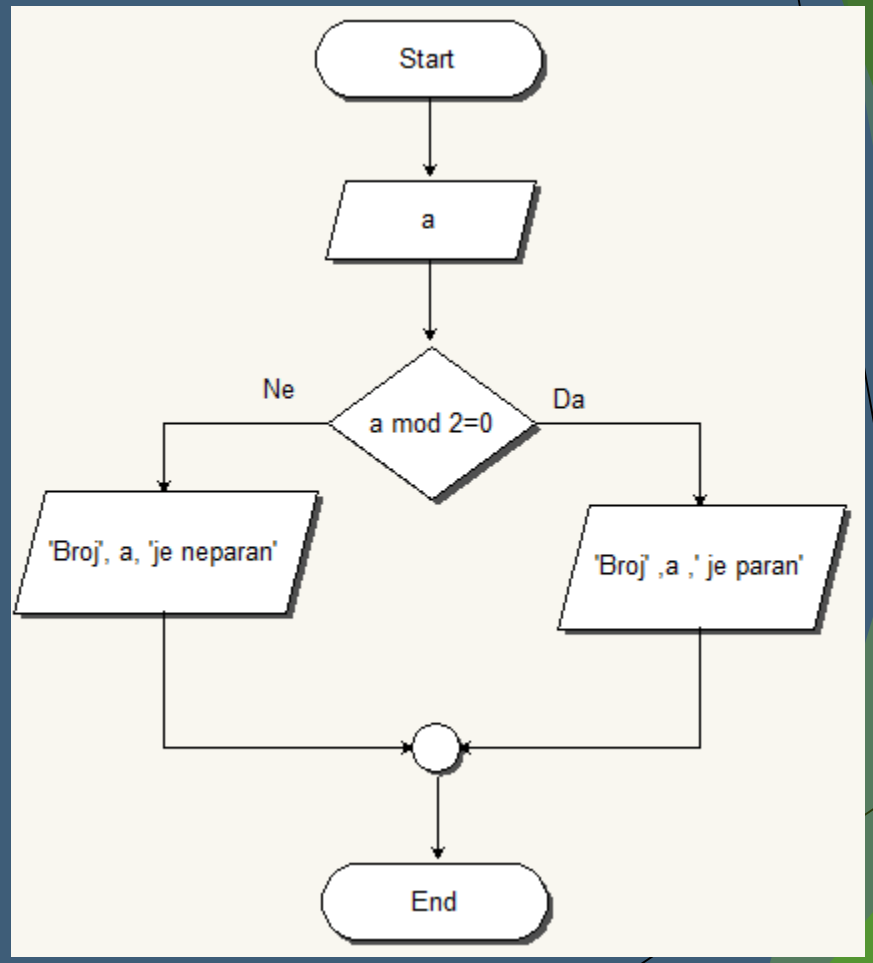
- Treba upisati paran broj a zatim ispisati da li je taj broj paran ili nije.
- Da bi broj bio paran on mora biti djeljiv sa brojem 2

(Kod djeljenja cijelih brojeva razlikujemo dvije vrste djeljenja:

div - računa cijeli dio djeljenja

mod – računa ostatak djeljenja

Znamo da je broj djeljiv drugim brojem ukoliko je ostatak djeljenja jednak 0)



Primjer 7 – veći/manji

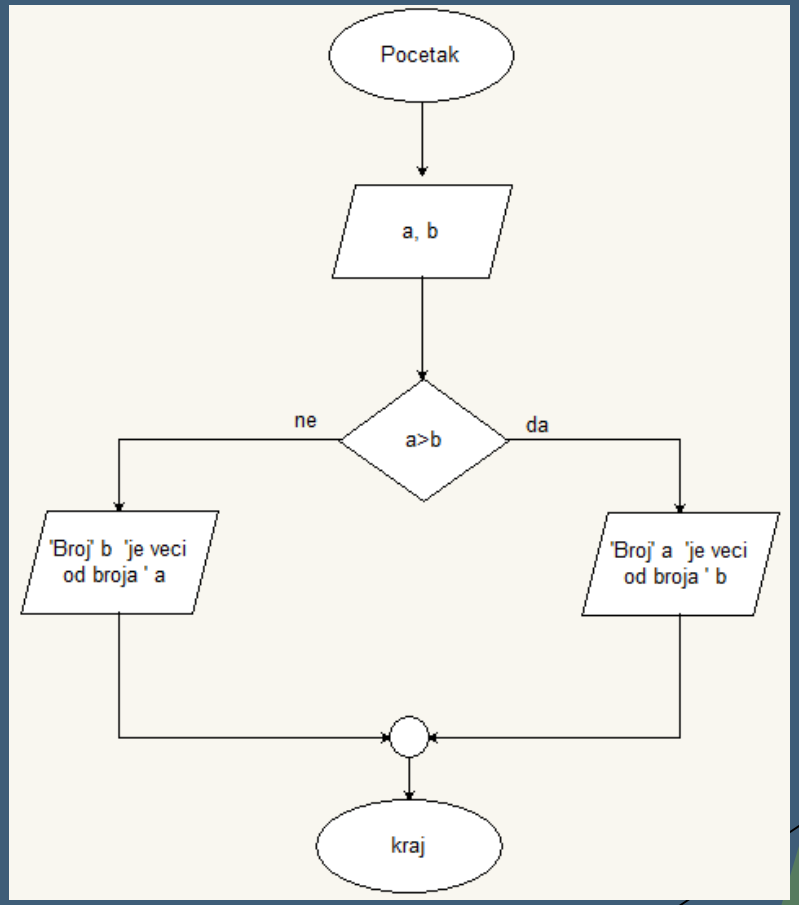
- Treba napraviti dijagram toka koji za dva unesena broja ispituje u kakvom su oni odnosu.

Jedno od mogucih rjesenja bi izgledalo kao na slici desno.

Ali sta ako korisnik unese dva broja 5?

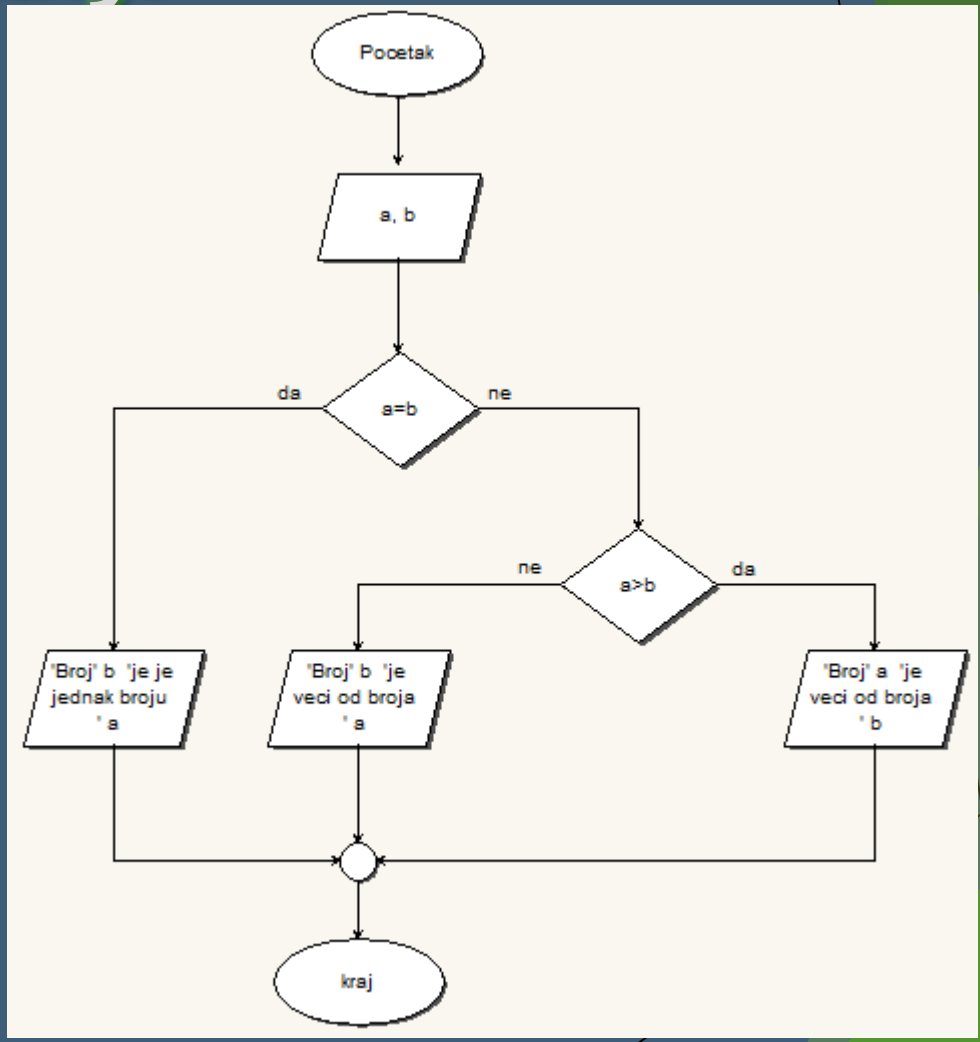
Nije tesko primjetiti da bi bila ispisana lijeva grana bloka grananja i imali bi:

„Broj 5 je veci od broja 5“



Primjer 7 – veći/manji

- Kako izbjeci situaciju sa prethodnog slajda?
- Jedno od rješenja bi bilo dodati novi blok grananja u kojem samo ispitujemo jednakost.

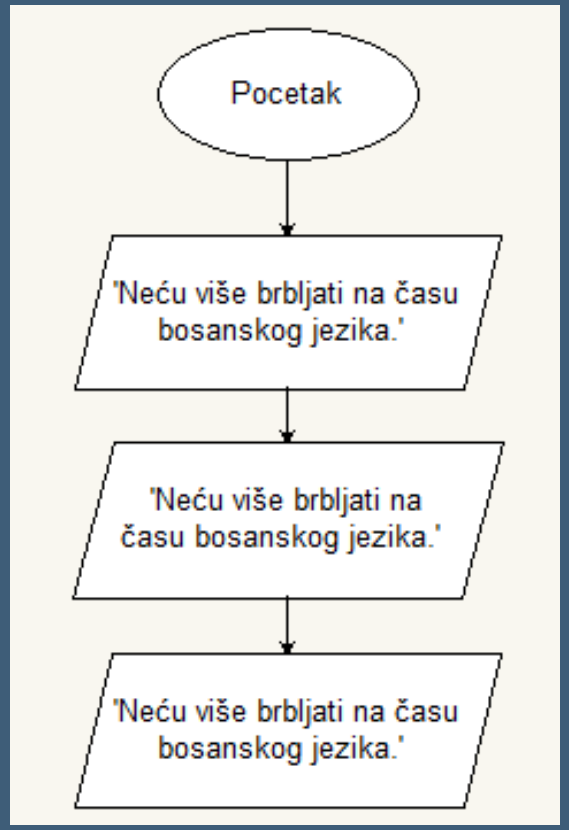


Petlje

- Dio programa, niz istih naredbi koje se ponavljaju dok je neki uvijet zadovoljen ili dok ne postane zadovoljen naziva se petlja.

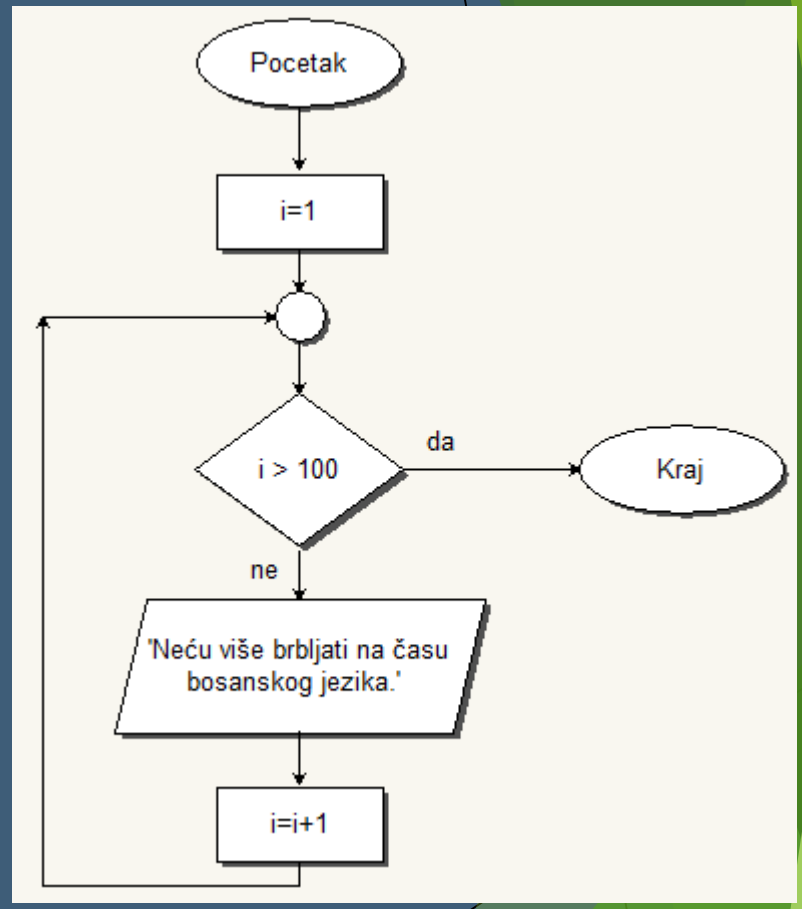
Primjer 8 – sto puta

- Pretpostavimo da ste na nastavi bili nemirni, recimo na času Bosanskog, hrvatskog i srpskog jezika i književnosti.
- Profesorica vam je za DZ zadala za kaznu da napišete sto puta:
- „Neću nikada više brbljati na času bosanskog jezika“
- Ako želite sebi da skratite posao mogli biste program riješiti grubom silom kao na sljedećem dijagramu:



Primjer 8 – sto puta

- Primjetimo da bi opet simbol za prikaz podataka sa rečenicom morali ili napraviti 100 puta ili kopirati. Kako god, opet bi to bilo ružno rješenje (ali ispravno).
- Uradimo elegantnije rješenje koristeći strukturu petlje.



Analizirajmo malo dijagram: Primjetimo da smo vrijednost varijable i postavili na 1, zatim ispitujemo da li je i veće od 100, pošto to nije slučaj idemo granom „ne“, ispisujemo jednom rešenicu a zatim varijabla i postaje 2 (i=1+1), vraćamo se u strukturu grananja, pošto je 2 i dalje manje od 100 izvršava se grana „ne“, ispisuje rečenica, varijabla i postaje 3 i ponovo se vraćmo u blok grananja. Primjetimo da će mo se vraćati tačno 100 puta.

Petlje

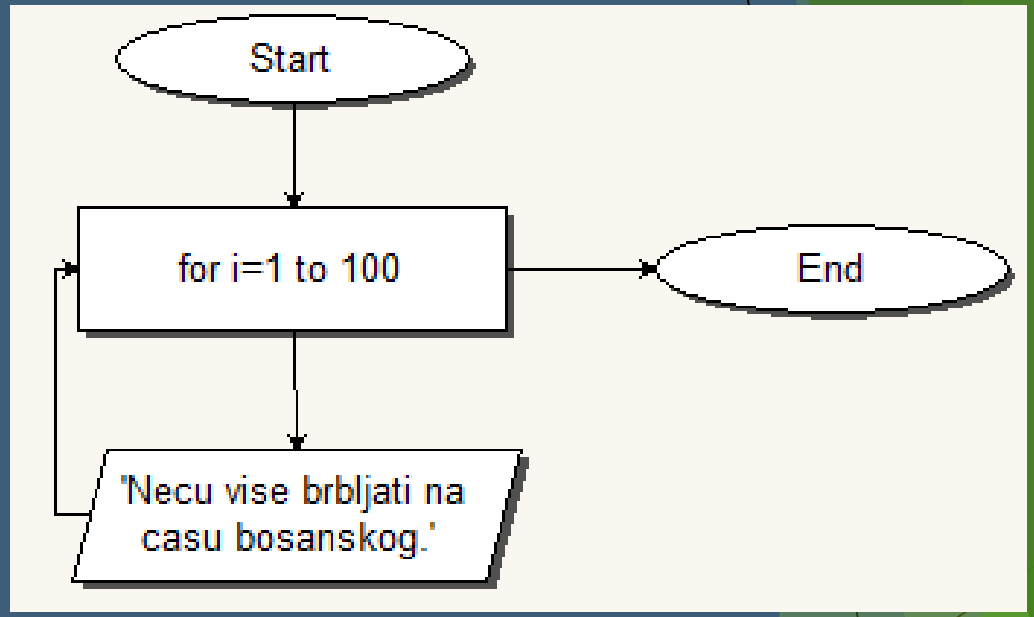
- Varijabla i iz prethodnog primjera se naziva **brojač** primjetimo da se ona na kraju slijeda (često kažemo na kraju iteracije) povečava za 1.

- Primjetimo da kada je varijabla i 100 puta prošla strukturom grananja njena će vrijednost postati 101, tada se više neće ići granom „ne“ nego će dijagram toka krenuti granom „da“ a tamo nailazi na znak end i algoritam završava (terminira)

- Na narednom slajdu pokazan je dijagram toka petlje koji je još kraći od prethodnog i koji će se koristiti često:

Primjer 8 – sto puta

- Oblik koji ćemo koristiti za ovaj oblik petlji

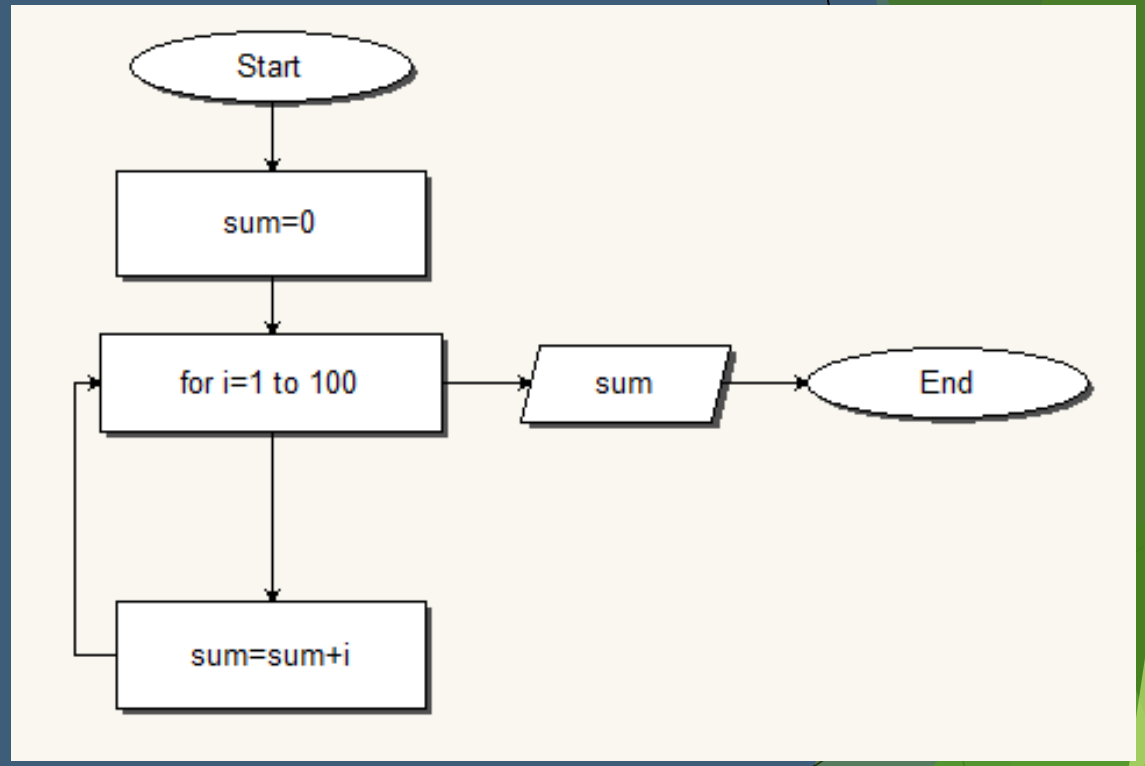


Primjer 9 – sabrati prvih 100 brojeva

- Zadatak nam je da saberemo prvih 100 brojeva. Koristeći petlje ovaj problem postaje pravo lagana stvar.
- Iz prethodnog primjera znamo da **for petlja** može povečava brojač u svakoj iteraciji.
- Napravit ćemo varijablu sum u koju ćemo u svakoj iteraciji petlje zbrajati brojač i, na kraju je potrebno varijablu **sum** ispisati.

Primjer 9 – sabrati prvih 100 brojeva

- Analizirajmo kratko zadatak, probajmo pratit vrijednosti varijable **sum**:
- U prvoj iteraciji **sum** postaje sum=0+1 dakle ima vrijednost 1
- U drugoj iteraciji je sum=1+2 dakle ima vrijednost 3
- U trecoj je sum=3+3 dakle sum=6
- U cetvrtoj je sum=6+4 dakle sum=10
- U petoj je sum=10+5=15
- .....
- Kada bi sabrali svih 100 brojeva onda bi se prekinula form petlja i ispisala bi se varijabla sum (to bi bio broj 5050)



Primjer 10 – racunanje prosjeka

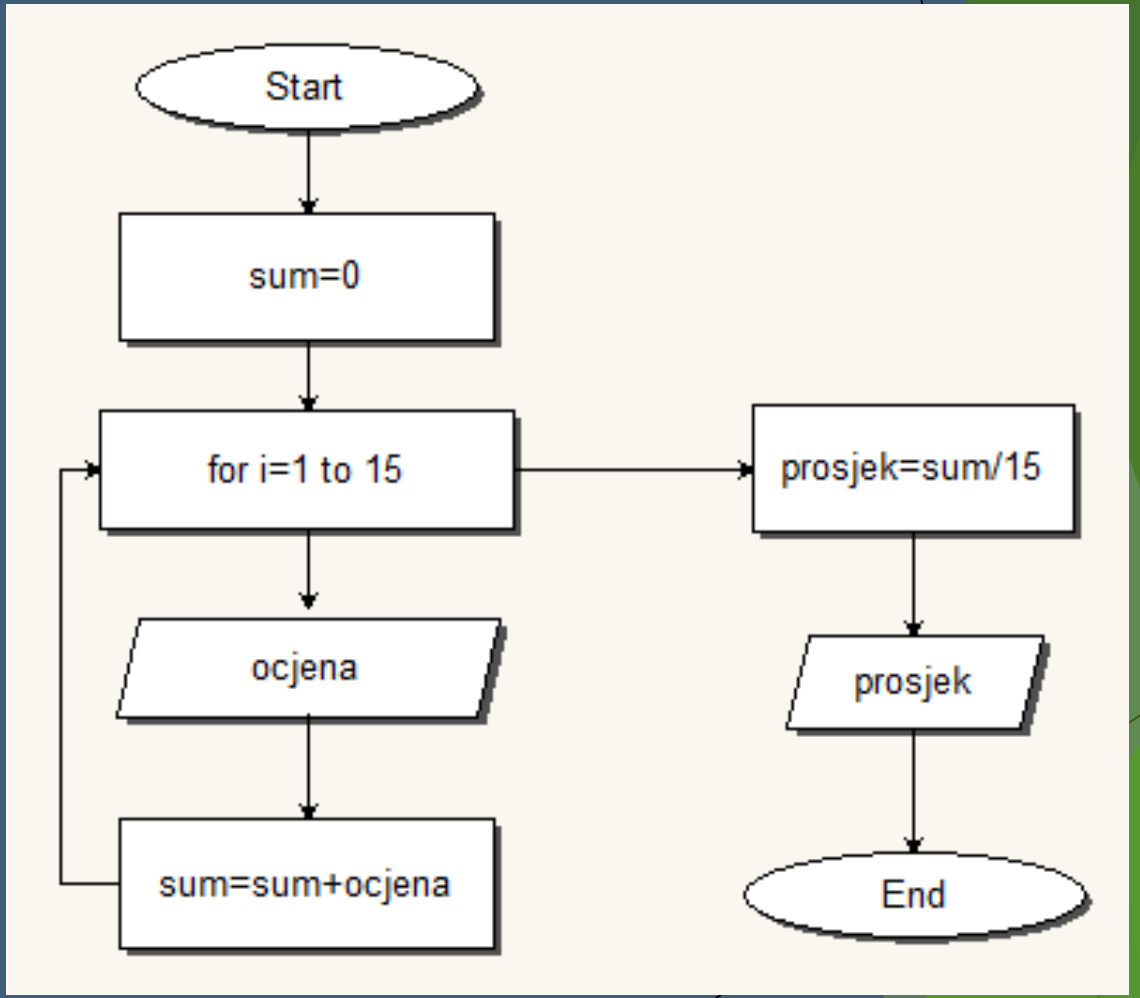
- Recimo da je došlo polugodište i da imate 15 predmeta. Ako želite da izračunate prosjek ocjena otprilike koristili bi se sljedećim metodom:

- - prvo bi sabrali sve zaključne ocjene po predmetima, dakle suma=predmet1+predmet2+.....+predmet15
- Zatim biste podjelili ovau sumu sa brojem predmeta, dakle Prosjek=suma/15

- Napravimo dijagram toka

Primjer 10 – racunanje prosjeka

- Dijagram prvo postavi vrijednost varijable sum na 0, zatim ulazimo u for petlju tu cemo prvo za i=1 da unesemo prvu ocjenu, zatim tu ocjenu sabiramo sa sadržajem varijable sum (to je 0) i tu vrijednost dodjeljujemo varijabli sum. Time je prva iteracija gotova.
- Sada i postaje 2, sve se ponovo izvršava kao kod prve iteracije.
- Kada se zavrsi 15-ta iteracija u varijabli su se nalazi sabrano 15 ocjena. For petlja se zavrsava. Dijagram racuna varijablu prosjek i zatim je ispisuje.



Petlje- DZ

- Zadatak 1

Treba da prepravite prethodni program tako se može unijeti koji broj predmeta imate
(Pošto neki učenici (ovisno od smjera) nemaju uvijek 15 predmeta, a sumnjam da će te ih i vi imati u II i III razredu.)

- Zadatak 2

Napraviti dijagram toka za program koji ispisuje samo parne brojeve od 1 do 99

Primjer 11 – zbir neparnih

- Napisati dijagram toka koji sabira neparne brojeve od nekog broja **n** pa do nekog broja **m**, pri čemu mora vrijediti **n<m**.

Kako znamo da je neki broj neparan?

- Iz matematike je poznato da kada broj dijelimo sa 2 možemo kao ostatak imati brojeve 0 ili 1. Ako je ostatak 0 to znači da je broj dijeljiv sa 2 odnosno da je paran. A ako je ostatak 1 onda je broj neparan.

Ostaje još problem – kako dobiti ostatak dijeljenja?

- Znamo od ranije da za obično djeljenje koristimo znak „ / “, ako želimo ostatak djeljenja koristit ćemo oznaku „ mod “

Primjeri:

7 mod 4 = 3

10 mod 3 = 1

5 mod 2 = 1

8 mod 2 = 0

Primjer 11 – zbir neparnih

- Nakon postavljanja varijable zbir na 0, unosimo vrijednosti **n** i **m** između kojoh se nalaze neparni brojevi koje želimo ispisati i sabrati.
- Primjetimo da brojač **i** u petlji sada broji od **n** do **m** , već smo rekli kako ispitujemo parnost dakle moramo koristiti blok grananja
- Ako broj koji ispitujemo nije neparan iteracija se prekida i prelazimo na sljedeću iteraciju, a ako jeste neparan onda taj broj ispišemo i zbrojimo njegovu vrijednost u varijablu **zbir**.
- Tek kada ispitamo sve vrijednosti između brojeva **n** i **m** dijagram izlazi iz for petlje, ispisje vrijednost varijable zbir i program terminira.