SVEUČILIŠTE U ZAGREBU
Fakultet prometnih znanosti
Zavod za inteligentne transportne sustave
Vukelićeva 4, Zagreb, HRVATSKA

Računalstvo

Dijagrami toka: grananje i petlje

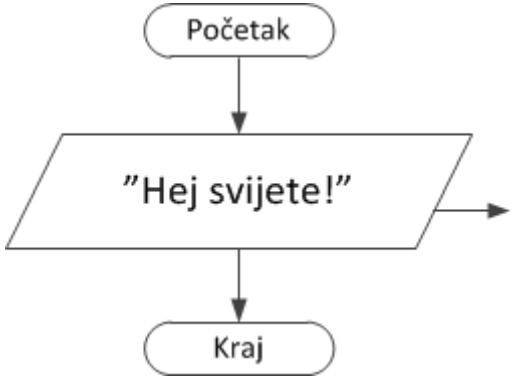
Doc. dr. sc. Edouard Ivanjko, dipl.ing.

Sadržaj

- Uvod
- Unos uz ispis poruke
- Ispis uz poruku
- Grananje
- Petlje
- Primjeri

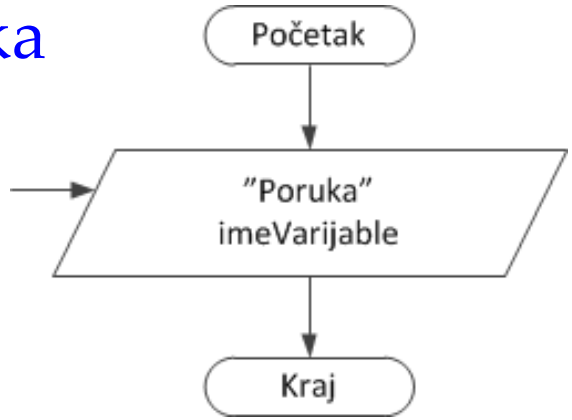
Uvod

- Dijagram toka olakšava kreiranje koncepta programa za rješenje zadanog problema
- Pojednostavljenja u radu s varijablama
  - Nije potrebna deklaracija varijable
  - Tip varijable se određuje automatski prilikom pridruživanja vrijednosti
- Sva pravila u pisanju izraza i obradi podataka sačuvana
  - Sintaksa ovisna o alatu
  - Varijable su nezavisne



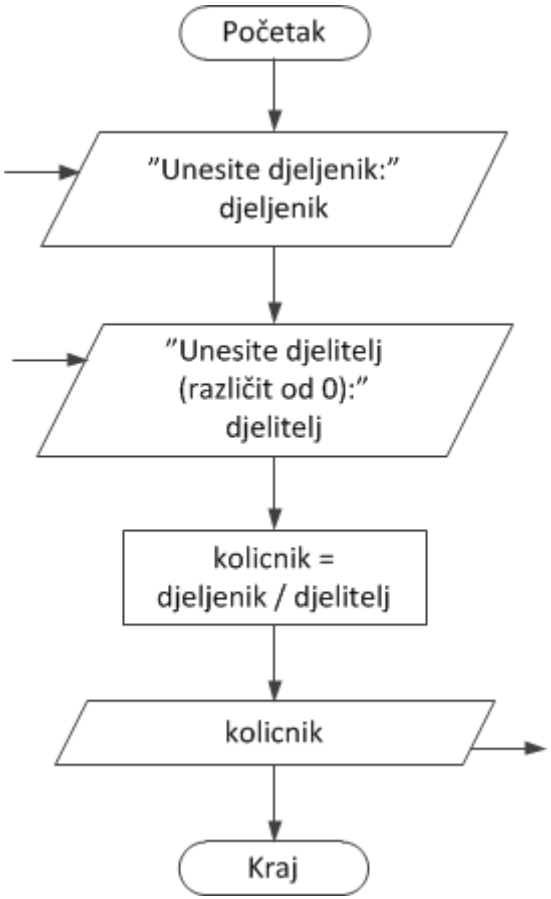
Unos uz ispis poruke

- Radi olakšanja unosa podataka se kreiraju forme
  - Navodi se poruka za objašnjenje traženog podatka
  - Operater vidi poruku te prostor za unos vrijednosti
  - Ime varijable bitno za dijagram toka, ne za operatera
- Blok dijagrama toka za unos podataka sadrži
  - Poruku objašnjena za operatera
  - Ime varijable za spremanje podatka



Unos uz ispis poruke – Primjer

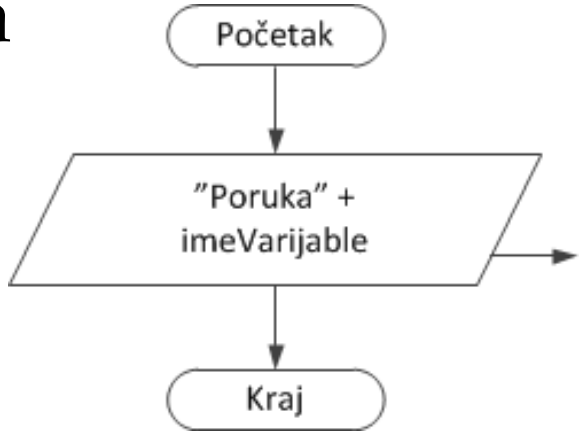
- Program za dijeljenje dva broja
  - Unos podataka sadrži pomoć za operatera
    - Poruka vodi operatera kroz izvršavanje programa
    - Olakšano izbjegavanje unosa krivih podataka za izvršavanje programa
  - Ime varijable operater ne vidi prilikom unosa



📂 **Dijeljenje**

Ispis uz poruku

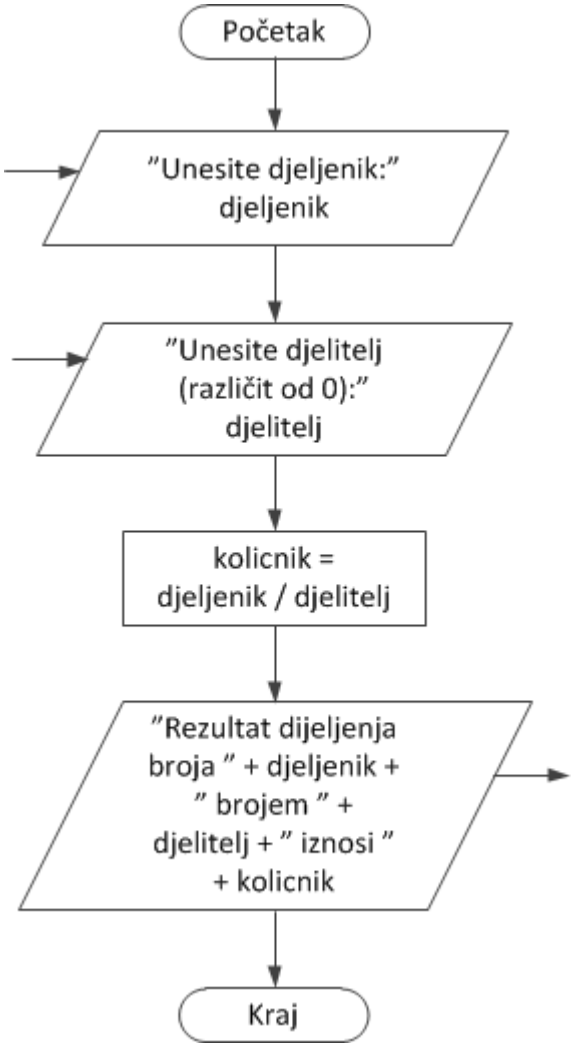
- Kod ispisa rezultata preporučljivo ispisati i poruku objašnjenja izračunatoga
- Ispis rezultata često sadrži
  - Poruku
  - Obrađene (ulazne) vrijednosti
  - Dobiveni rezultat



- Za kreiranje izlaznih poruka koristi se operator nadovezivanja „+" radi spajanja niza znakova i vrijednosti varijable (tzv. proces konkatenacije)
  - Na zaslonu računala se vidi ispis kao cjelina poruke i vrijednosti varijable

Ispis uz poruku – Primjer

- Program za dijeljenje dva broja
  - Ispis rezultata programa sada također sadrži pripadnu poruku
    - Bitno poštivati imena varijabli
    - Svako spajanje niza znakova i vrijednosti varijable se radi operatorom „+”
      - Kod ispisa spaja nizove znakova
  - Kod ispisa se dohvaćaju trenutne vrijednosti varijabli
    - Automatska pretvorba sadržaja varijable u niz znakova



`DijeljenjeIspis`

Grananje

- Grananje programa omogućuje kreiranje više smjerova izvršavanja programa
  - Moguće odabrati samo jedan smjer
  - Svi smjerovi se na kraju spajaju u jedan prije bloka završetka
- Moguće implementirati 3 vrste grananja
  - Grananje uz izvršavanje naredbe samo ako je uvjet ispunjen
  - Grananje uz naredbe za oba slučaja ispunjenosti uvjeta
  - Grananje uz više uvjeta

Grananje – Naredba ako je uvjet ispunjen

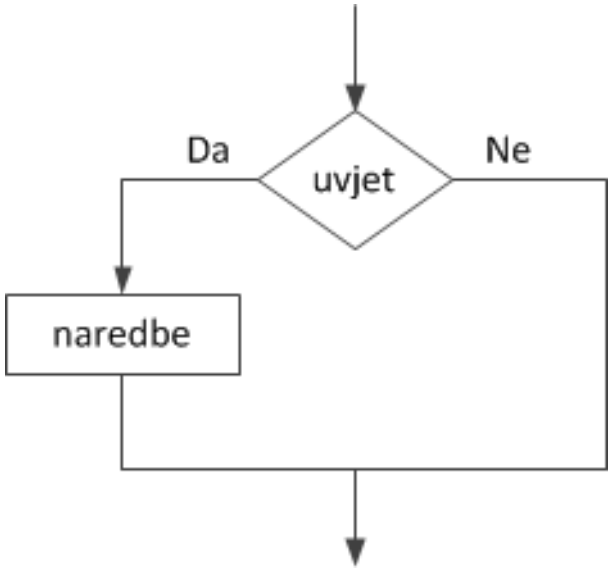
- Koristi se samo DA grana bloka za provjeru uvjeta (grananje if)
  - Druga NE grana se crta bez naredbe
    - Obavezno se prikazuje zbog pravila izgleda bloka za provjeru uvjeta
- Jednostavniji programi kada je potreban utjecaj na izvršavanje programa samo u specifičnom slučaju
  - Poruka vozaču kada je prekoračio dopuštenu brzinu, otvaranje vrata parkirališta kada se pojavi korisnik, aktiviranje rasvjete/nadzorne kamere kada se pojavi čovjek/incident, …

Grananje – Naredba ako je uvjet ispunjen

- Pseudokôd
  - Nema dijela vezan uz „inače”

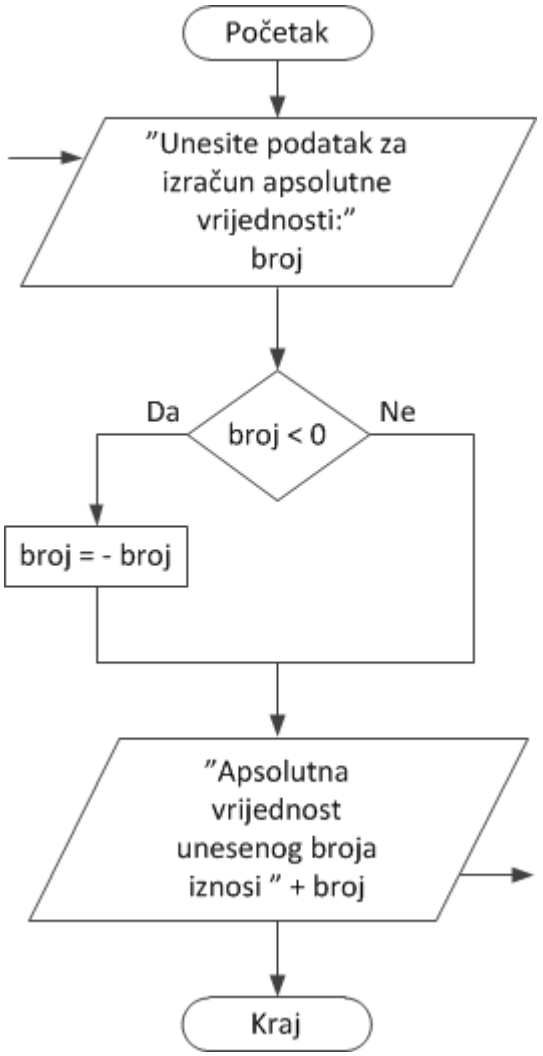
```
ako je uvjet onda
{
   naredbe
}
```

- Dijagram toka
  - Nakon izvršavanja bloka usporedbe i odabrane grane radi se spajanje
    - Opet se dobiva jedan smjer izvršavanja dijagrama toka



Grananje – Naredba ako je uvjet ispunjen

- Primjer
  - Izračun apsolutne vrijednosti unesenog broja
    - Promjena unesene vrijednosti se radi samo ako je ona negativna
      - Potrebno je promijeniti predznak
    - U suprotnom je uneseni broj ujedno i njegova apsolutna vrijednost



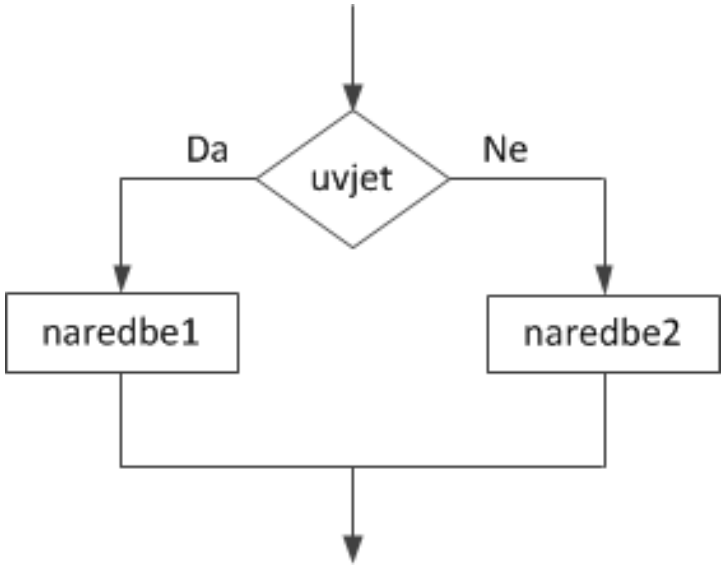
`ApsolutnaVrijednost`

Grananje – Naredbe za oba slučaja uvjeta

- Koriste se obje grane bloka za provjeru uvjeta (grananje if – else)
- Pseudokôd
  - Koristi se i naredba uz dio „inače”

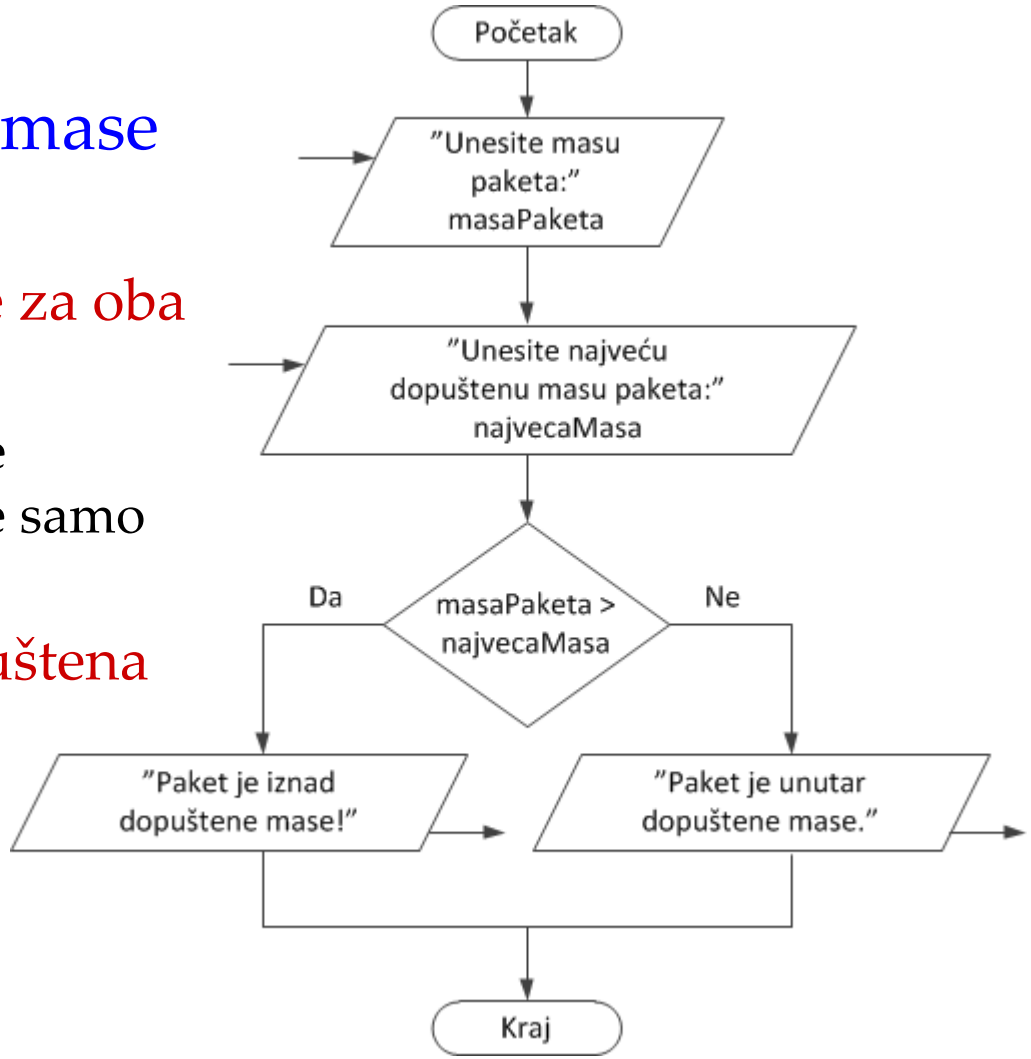
```
ako je uvjet onda
      naredba1
inače
      naredba2
```

- Dijagram toka
  - Moguće u svakoj grani dati jednu ili više naredbi



Grananje – Naredbe za oba slučaja uvjeta

- Primjer
  - Program za provjeru mase paketa
    - Ispis pripadne poruke za oba slučaja
      - Samo jedna grana se izvršava i ispisuje se samo jedna poruka
    - Unesena najveća dopuštena masa je uključena u dozvoljeno područje



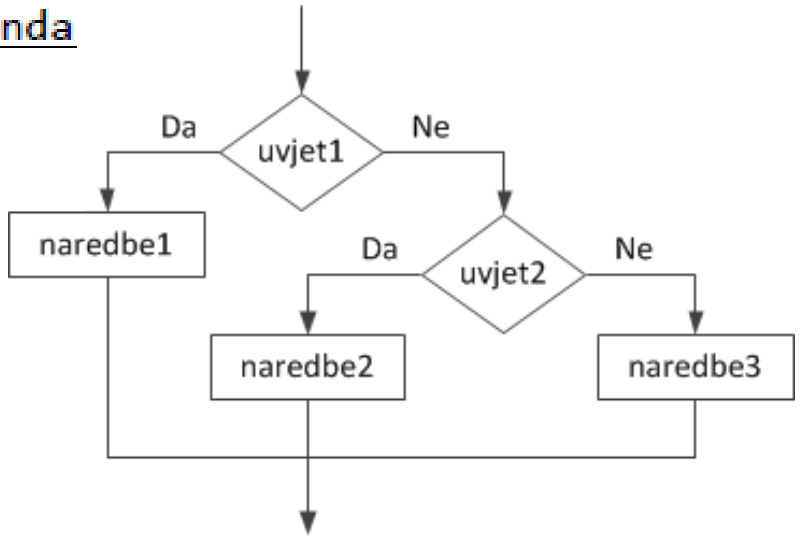
📂 **MasaPaketa**

Grananje – Ispitivanje više uvjeta

- Koristi se slijedno ispitivanje uvjeta (grananje if – else if – else)
  - Sljedeći uvjet se ispituje ukoliko trenutni nije ispunjen
  - Završetak grananja čim je jedan uvjet ispunjen
    - Nakon izvođenja pripadne naredbe
- Pseudokôd

```
ako je uvjet1 onda
        naredba1
inače ako je uvjet2 onda
        naredba2
inače
        naredba3
```

- Dijagram toka
  - Na kraju grananja spajanje i nastavak u jednom smjeru



Grananje – Ispitivanje više uvjeta

- Primjer
  - Program za ispitivanje prekoračenja dozvoljene brzine
    - Dopušteno odstupanje (prekoračenje) od 10%
  - Za program su potrebne dvije varijable

Ime varijable	Tip varijable	Značenje varijable
brzina	cijeli broj	Trenutna brzina vozila
najvecaBrzina	cijeli broj	Najveća dopuštena brzina

Grananje – Ispitivanje više uvjeta

- Primjer
  - Program za ispitivanje prekoračenja dozvoljene brzine
    - Dopušteno odstupanje (prekoračenje) od 10%
    - Problem je moguće riješiti pomoću grananja
      - Ispitivanje više uvjeta
    - Pseudokôd

Unos

brzina, najvecaBrzina

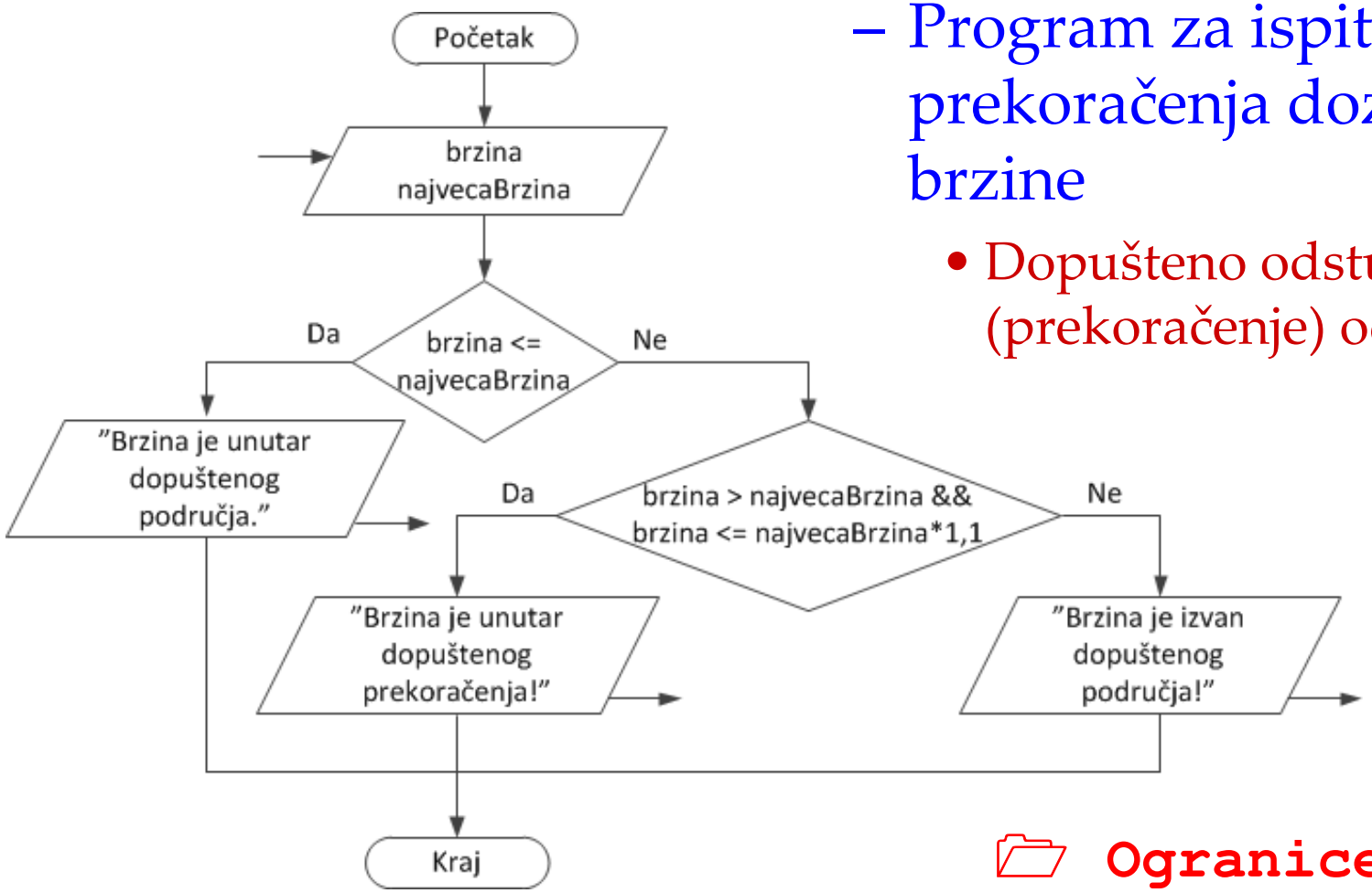
Grananje – Ispitivanje više uvjeta

- Primjer
  - Program za ispitivanje prekoračenja dozvoljene brzine
    - Dopušteno odstupanje (prekoračenje) od 10%
    - Pseudokôd

```
ako je brzina <= najvecaBrzina onda
    Ispis
        „Brzina je unutar dopuštenog područja.”
inače ako je brzina > najvecaBrzina &&
        brzina <= najvecaBrzina * 1,1 onda
    Ispis
        „Brzina je unutar dopuštenog prekoračenja!”
inače
    Ispis
        „Brzina je izvan dopuštenog prekoračenja!”
```

Grananje – Ispitivanje više uvjeta

- Primjer
  - Program za ispitivanje prekoračenja dozvoljene brzine
    - Dopušteno odstupanje (prekoračenje) od 10%



📂 `OgranicenjeBrzine`

Petlje

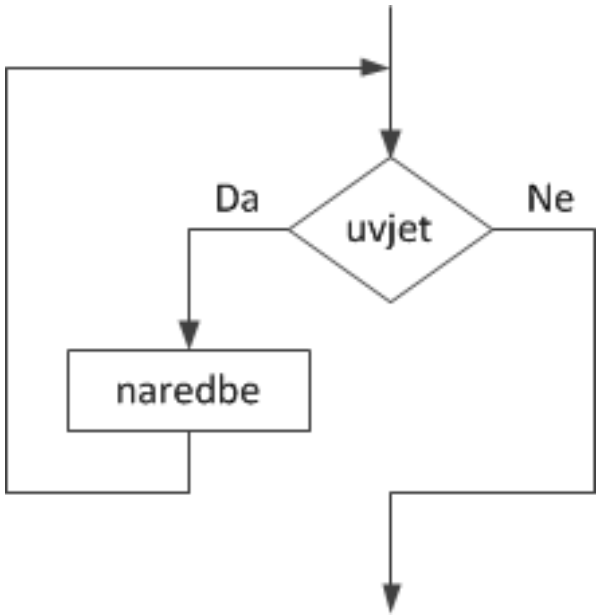
- Omogućuju primjenu istog skupa naredbi (tijela petlje) na drugoj vrijednosti podatka
- Blokovi dijagrama toka omogućuju implementaciju dvije vrsti petlji
  - Provjera uvjeta prije izvršavanja tijela petlje (petlja while)
  - Provjera uvjeta nakon izvršavanje tijela petlje (petlja do – while)
- Ne postoji blok za automatsku promjenu brojila količine obrađenih podataka
  - Potrebno kod petlje s poznatim brojem izvršavanja (petlja for)
    - Ta vrste petlje se može samo ručno implementirati

Petlje – Ispitivanje uvjeta prije izvršavanja

- Tijelo petlje će se izvršiti samo ako je uvjet ispunjen
  - Nakon izvršavanja tijela petlje ponovno se izvršava provjera uvjeta
- Pseudokôd

```
dok je uvjet činiti
    naredba
```

- Dijagram toka



- Naziva se i petlja „while”

Petlje – Ispitivanje uvjeta prije izvršavanja

- U Raptor-u se tijelo petlje izvršava kada uvjet nije ispunjen
- Prilikom implementacije potrebno prilagoditi uvjet
  - Koristi se negacija uvjeta
  - Uvjet **„i > 0"** koji izvršava tijelo petlje kada je istinit analogan uvjetu **„i <= 0"** koji izvršava tijelo petlje kada nije istinit
    - Područje vrijednosti varijable **„i"** kada se tijelo petlje izvršava ostaje isto

Petlje – Ispitivanje uvjeta prije izvršavanja

- Primjer
  - Izračun aritmetičke sredine n brojeva
  - Za ovaj program su potrebne četiri varijable

Ime varijable	Tip varijable	Značenje varijable
n	cijeli broj	Količina vrijednosti za izračun aritmetičke sredine
i	cijeli broj	Količina do sada obrađenih podataka
zbroj	cijeli broj	Zbroj do sada obrađenih podataka
sredina	broj s plivajućim zarezom	Aritmetička sredina obrađenih podataka

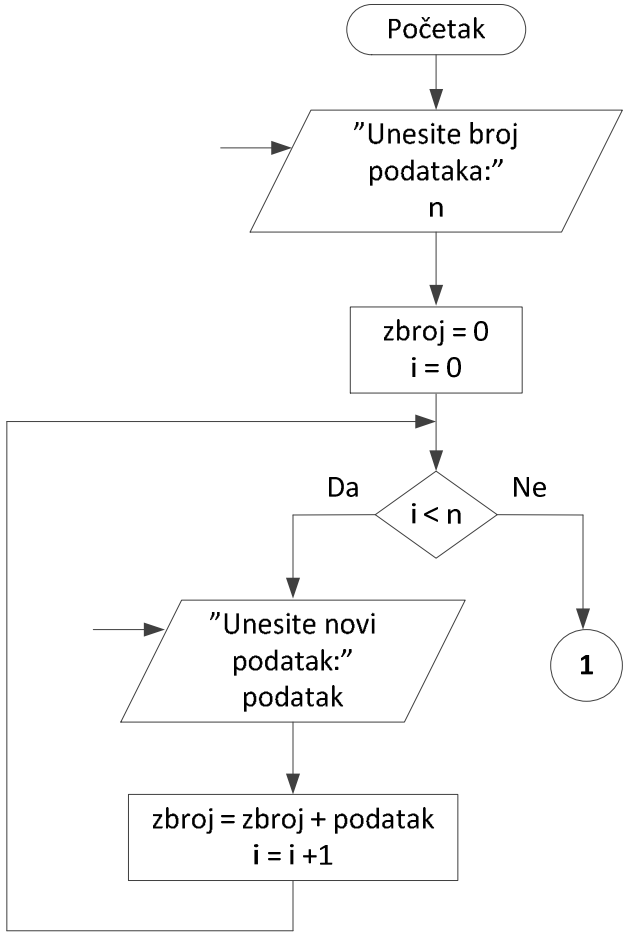
Petlje – Ispitivanje uvjeta prije izvršavanja

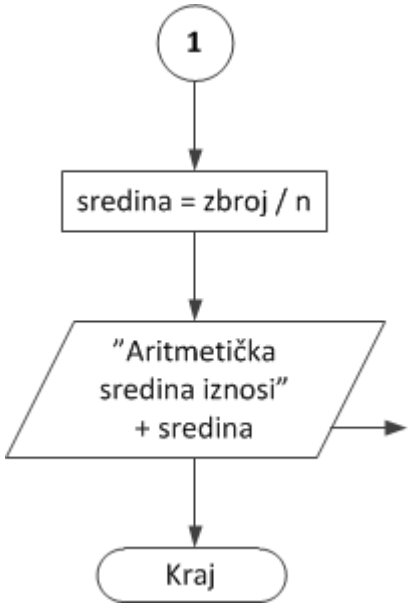
- Primjer
  - Izračun aritmetičke sredine n brojeva
    - Pseudokôd

```
Inicijaliziraj
    zbroj := 0
    i := 0
Unos
    n
dok je i < n činiti
    Unos
        podatak
    Izračunaj
        zbroj := zbroj + podatak
        i := i +1
Izračunaj
    sredina := zbroj / n
Ispis
    sredina
```

Petlje – Ispitivanje uvjeta prije izvršavanja

- Primjer
  - Izračun aritmetičke sredine n brojeva





AritmetickaSredina

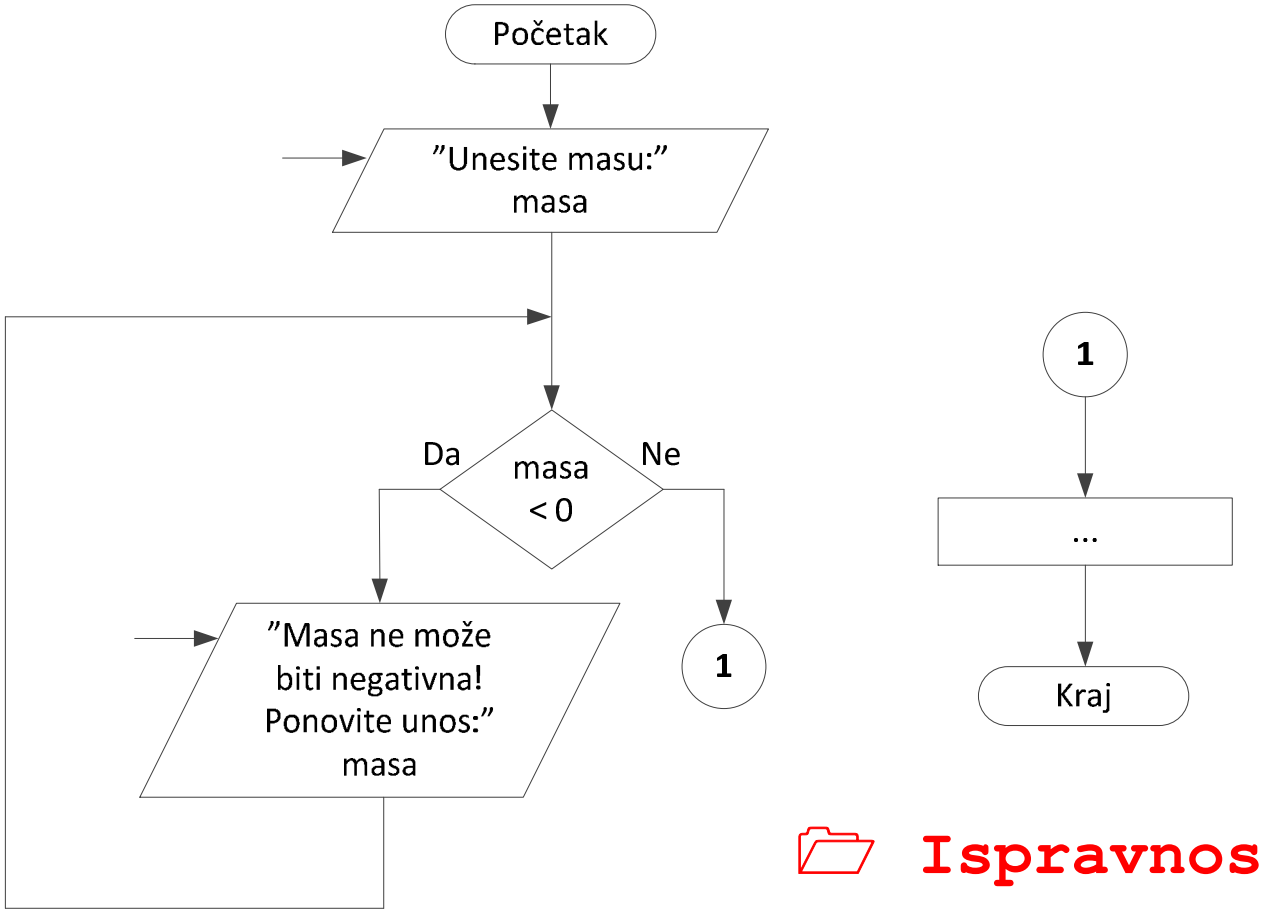
Petlje – Ispitivanje uvjeta prije izvršavanja

- Primjer
  - Ispitivanje ispravnosti unesenog podatka mase
    - Pseudokôd

```
Unos
    masa
dok je masa < 0 činiti
    Ispis
        "Masa ne može biti negativna! Ponovite unos:"
    Unos
        masa
Izračunaj
    …
```

Petlje – Ispitivanje uvjeta prije izvršavanja

- Primjer
  - Ispitivanje ispravnosti unesenog podatka mase



📂 **IspravnostUnosaMase**

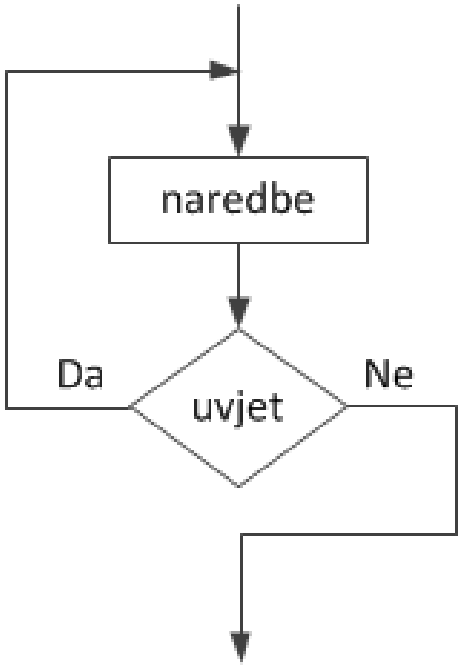
Petlje – Ispitivanje uvjeta nakon izvršavanja

- Tijelo petlje će se izvršiti najmanje jednom
  - Uvjet se ispituje tek nakon izvršavanja tijela petlje

- Pseudokôd

```
činiti
   naredba
dok je uvjet
```

- Dijagram toka



- Naziva se i petlja „do while”

Petlje – Ispitivanje uvjeta nakon izvršavanja

- Primjer
  - Učitavati podatke u program dok operater ne unese podatak 0
    - Pseudokôd

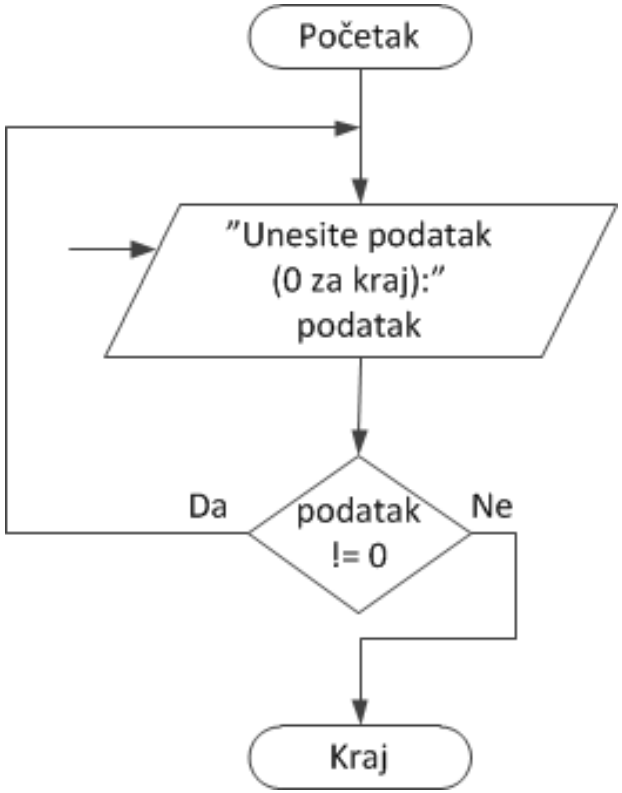
činiti

Unos

podatak

dok je podatak != 0

…



📂 **IspravnostUnosa**

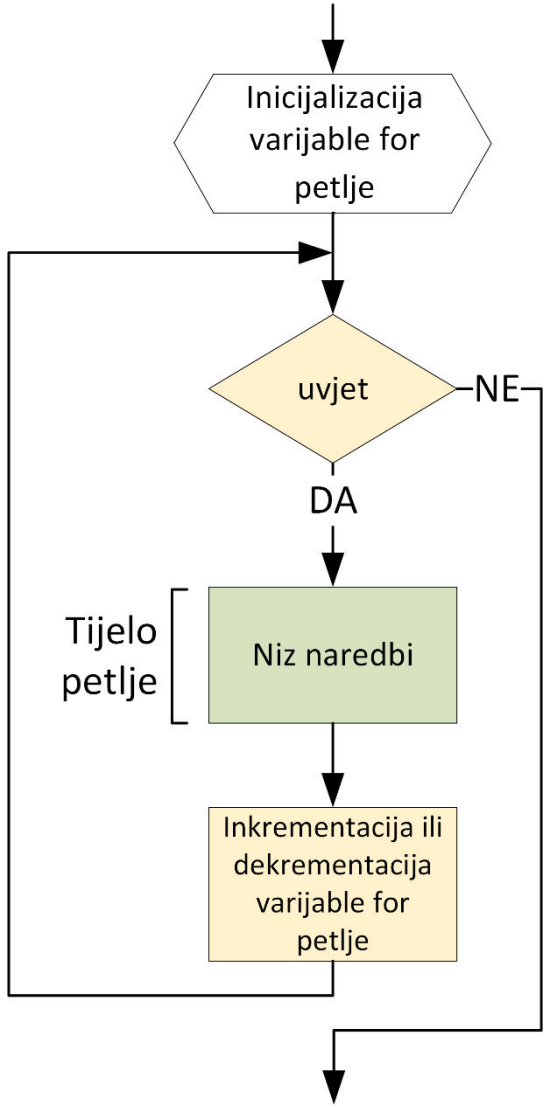
Petlje – Poznati broj izvršavanja

- Prednost ove petlje što se po definiciji automatski radi promjena kontrolne varijable
  - Varijabla u uvjetu se automatski inkrementira (uvećava) ili dekrementira (umanjuje)
  - U dijagramu toka se ova petlja ručno implementira korištenjem osnovnih blokova
- Ova petlja se naziva i petlja for
- Općeniti pseudokôd

```
za b := p do n korak k činiti
{
    naredbe
}
```

Petlje – Poznati broj izvršavanja

- Formalni koraci petlje for
  - Inicijalizacija varijabli
    - Izvršava se samo jednom
  - Provjera uvjeta
  - Izvršavanje tijela petlje
    - Samo ako uvjet ispunjen
  - Promjena kontrolne varijable
    - Samo ako je tijelo petlje izvršeno
  - Provjera uvjeta
- Ako uvjet nije ispunjen izvršava se prva naredba nakon petlje



Petlje – Poznati broj izvršavanja

- Primjer
  - Potrebno je izračunati zbroj cijelih brojeva između donje i gornje granice intervala s time da su obje granice uključene
  - Za program su potrebne četiri varijable

Ime varijable	Tip varijable	Značenje varijable
i	cijeli broj	Količina do sada zbrojenih brojeva
donjaGranica	cijeli broj	Donja granica intervala
gornjaGranica	cijeli broj	Gornja granica intervala
zbroj	cijeli broj	Zbroj cijelih brojeva unutar intervala

Petlje – Poznati broj izvršavanja

- Primjer
  - Potrebno je izračunati zbroj cijelih brojeva između donje i gornje granice intervala s time da su obje granice uključene
  - Pseudokôd

```
Inicijaliziraj
        zbroj := 0
Ispis
        ”Unesite vrijednost donje granice intervala >”
Unos
        donjaGranica
```

Petlje – Poznati broj izvršavanja

- Primjer
  - Potrebno je izračunati zbroj cijelih brojeva između donje i gornje granice intervala s time da su obje granice uključene
  - Pseudokôd

```
Ispis
    "Unesite vrijednost gornje granice intervala >"
Unos
    gornjaGranica
za i=donjaGranica do gornjaGranica činiti
    zbroj := zbroj + 1
Ispis
    "Zbroj cijelih brojeva unutar intervala iznosi " + zbroj
```

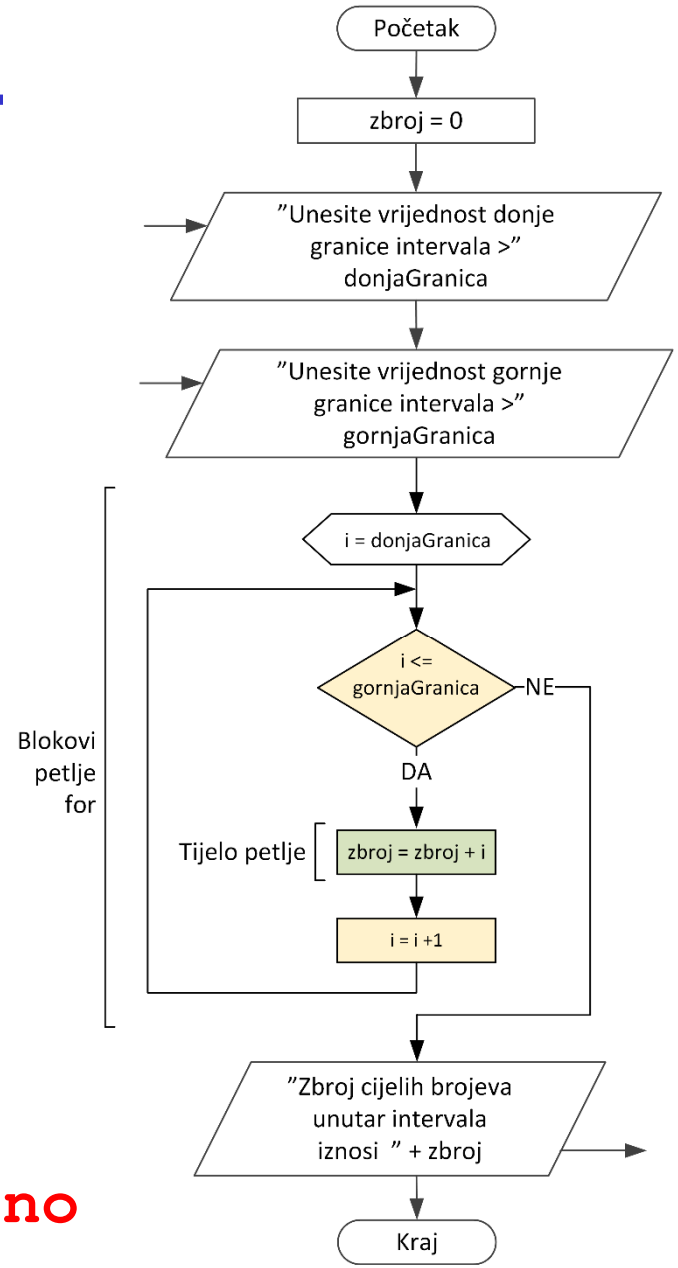
Petlje – Poznati broj izvršavanja

- Primjer
  - Potrebno je izračunati zbroj cijelih brojeva između donje i gornje granice intervala s time da su obje granice uključene
    - Provjera testnim podacima

      donjaGranica = 5

      gornjaGranica = 10

      - Točan zbroj iznosi 45



📂 **ZbrojIntervalUkljuceno**