Algoritmi i programiranje

Osnovni pojmovi i načini predstavljanja algoritama

Etape u rešavanju problema uz pomoć računara

1. Precizan opis problema
2. Definisanje matematičkog modela i izbor metoda za njegovo rešavanje
3. Projektovanje algoritama
4. Pisanje i testiranje programa
5. Izrada prateće dokumentacije

1. Opis problema

- definisati koje su informacije poznate i dostupne kao ulazni podaci
- koja informacija i u kom obliku treba da se dobije kao rezultat
- u ovu etapu mogu biti uključena i lica koja se ne bave programiranjem (lekari, tehnolozi, ekonomisti,...)

2. Definisanje modela i izbor metoda

- Nakon definisanja problema, prelazi se na opis problema formalnim, matematičkim aparatom korišćenjem matematičkih formula i relacija
  - Primer:
    - Sistem linearnih j-na je $A\vec{x} = \vec{b}$ mat. model
    - treba odabrati metod za nalaženje nepoznatog vektora x

$$x_i = \frac{\det A_i}{\det A} \quad \text{(Kramerov metod)}$$

$$\vec{x} = A^{-1}\vec{b}$$

$$\text{po definiciji} \quad A^{-1} = \frac{adjA}{\det A}$$

Ovaj način se skoro nikad ne koristi, već LU, Gausov metod, Keli-Hamiltonova teorema,...

2. Definisanje modela i izbor metoda

- Pored tvz. tačnih metoda za rešavanje, postoje i približne metode
- U računarstvu se veoma često koristi aparat numeričke analize (jer se zahteva dobijanje nekog numeričkog reazultata)
  - Postoji čitav niz problema koji se ne mogu rešiti klasičnim metodama ili je njihovo rešavanje isuviše glomazno
    - npr. za rešavanje sistema j-na velikog reda isključivo se koriste numerički metodi
    - numerički metodi koriste samo osnovne aritmetičke operacije

Primer

- Naći pozitivan koren j-ne $x^2-a=0$, $a>0$
- Traženo egzaktno rešenje je $x = \sqrt{a}$
- ako nam je potrebna brojna vrednost, ovo rešenje nije od koristi
- Jedan numerički metod za rešavanje j-ne $x^2-a=0$, svodi se na izračunavanje niza $\{X_k\}$:

$$x_0 = a, \quad x_{k+1} = \frac{1}{2}\left(x_k + \frac{a}{x_k} \right), \quad k = 0,1,2,...$$

Može se pokazati da niz $\{X_k\}$ konvergira ka $\sqrt{a}$, kada $k \to \infty$

3. Projektovanje algoritma

- U Websterovom rečniku se reč **algoritam** definiše kao specijalni metod za rešavanje neke klase problema.
  - u računarskoj tehnici, reč algoritam se odnosi na precizan metod koji se može iskoristiti za rešavanje problema na računaru
    - *geneza reči algoritam*: predstavlja latiniziranu transkripciju imena poznatog naučnika iz IX veka iz Srednje Azije, Muhameda Al-Horezma, tj. Muhameda iz Horezma
    - Horezmo je stara država koja se nalazila na teritoriji današnjeg Uzbekistana
    - Muhamed Al-Horezmi je, izmedju ostalog, definisao niz postupaka (pravila) za razna izračunavanja

Osobine valjanog algoritma

- Diskretnost
- Determinisanost
- Efektivnost (konačnost)
- Rezultativnost
- Masovnost
- (Optimalnost)

Osobine valjanog algoritma

- Diskretnost
  - algoritam se sastoji od konačnog broja posebnih koraka (algoritamski koraci)
  - svaki korak može zahtevati obavljanje jedne ili više operacija
  - u zavisnosti od toga koje operacije računar može da obavi, uvode se ograničenja za tip operacija koje se u algoritmu mogu koristiti (najčešće +, -, *, /)

Osobine valjanog algoritma

- Determinisanost
  - svaki algoritamski korak mora biti strogo definisan i potpuno jasan
    - npr. izraz izračunaj 5/0 ili “dodaj 6 ili 7 x” nisu dozvoljeni
  - posle izvršenja tekućeg alg. koraka mora biti jednoznačno definisano koji je sledeći korak

Osobine valjanog algoritma

- Efektivnost (konačnost)
  - sve operacije koje se javljaju u jednom algoritamskom koraku moraju se izvršiti za konačno (razumno kratko) vreme
  - vreme izvršenja celog algoritma mora biti konačno, tj. razumno dugo
- Rezultativnost
  - svaki algoritam mora posle konačnog broja koraka generisati traženi rezultat
  - algoritam može imati 0 ili više ulaznih podataka i može generisati jedan ili više izlaznih rezultata

Osobine valjanog algoritma

- Masovnost
  - svaki algoritam definiše postupak za rešavanje klase problema, a ne pojedinačnog slučaja
    - npr. rešavanje sistema j-na bilo kog reda, množenje matrica proizvoljnog reda,...

4. Pisanje i testiranje programa

- Nakon razrade algoritma prelazi se na pisanje programa
  - program mora biti napisan u formi koju računar "razume"
  - svaki algoritamski korak se zamenjuje nizom instrukcija (instrukcije čine računarski program)
    - svaka instrukcija mora biti napisana po odredjenim pravilima. Ta pravila čine jezik kojim se izdaju naredbe računaru (programski jezik)
    - svaki programski jezik ima skup pravila kojima se definišu važeće jezičke konstrukcije – sintaksa jezika
    - značenje (dejstvo) instrukcija čini semantiku jezika

4. Pisanje i testiranje programa

- Jedini jezik direktno razumljiv računaru je mašinski jezik
  - za računar ovaj jezik predstavlja niz elektronskih impulsa
  - programer ovaj jezik simbolički izražava pomoću binarnih brojeva (niz nula i jedinica)
    - svaki računar ima svoj mašinski jezik koji je direktno zavisan od hardvera računara (zovu se još i niži programski jezici)

4. Pisanje i testiranje programa

- Viši programski jezici su nezavisni od hardvera računara.

  – Program napisan na višem programskom jeziku za jedan računar, može se izvršavati i na drugom računaru

  – Program napisan na višem programskom jeziku računar ne može direktno "razumeti".

  – Potreban mu je prevodilac sa višeg na niži jezik.

  – Prevodjenje obavljaju posebni programi – kompilatori (kompajleri, prevodioci)

  – U fazi prevodjenja programa vrši se sintaksna analiza i otkrivaju se formalne (pravopisne) greške.

  – greške logičkog tipa računar ne može otkriti

4. Pisanje i testiranje programa

- Nakon faze kompajliranja, prelazi se na izvršenje sa test primerima
  - testiranje se obavlja za poznate rezultate
  - testiranjem se mogu otkriti logičke greške
  - test primere treba odabrati tako da se kroz svaku granu algoritma, tj. liniju programa prodje bar jednom
  - Testiranjem se ne dokazuje korektnost, već neispravnost algoritma ili programa
  - Za dokaz korektnosti programa koristi se aparat matematičke logike

5. Izrada prateće dokumentacije

- Dokumentacija sadrži
  - opis problema
  - korišćeni metod
  - program
  - uputstvo za instaliranje i korišćenje programa

Načini predstavljanja algoritama

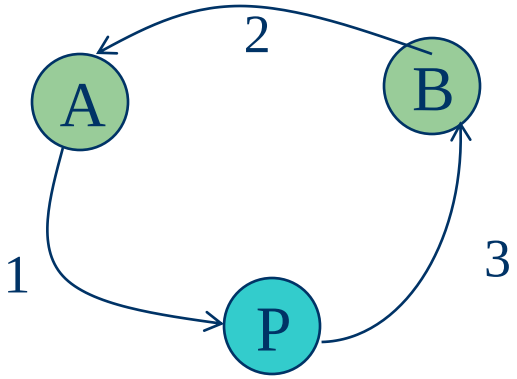
- Tekstualni
- Grafički (pomoću blok šema)
- Pseudo kodom
- Strukturogramom

Tekstualni način

- Koriste se precizne rečenice govornog jezika
  - Koristi se za lica koja se prvi put sreću sa pojmom algoritma
  - Dobra osobina: razumljivost za širi krug ljudi
  - Loše: nepreciznost koja proističe iz nepreciznosti samog jezika

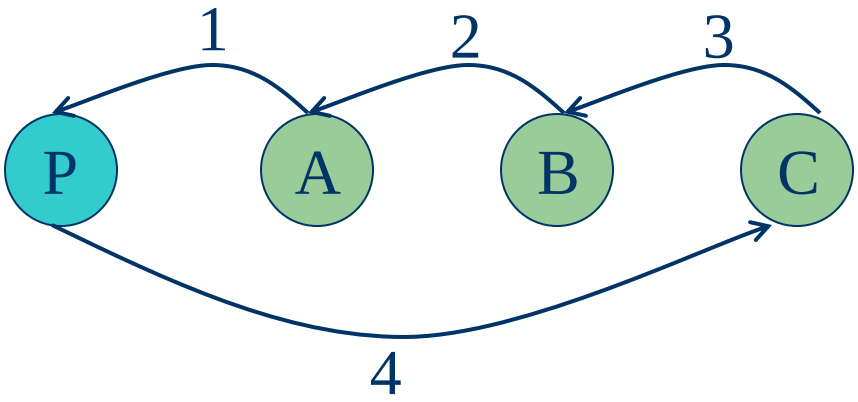
Primer 1

- Zameniti sadržaje dve memorijske lokacije, A i B
  - Rešenje – potrebna je treća, pomoćna, lokacija
    1. sadržaj lokacije A zapamtiti u pomoćnu lokaciju, P
    2. sadržaj lokacije B zapamtiti u lokaciju A
    3. sadržaj lokacije P zapamtiti u lokaciju B
    4. kraj



Primer 2

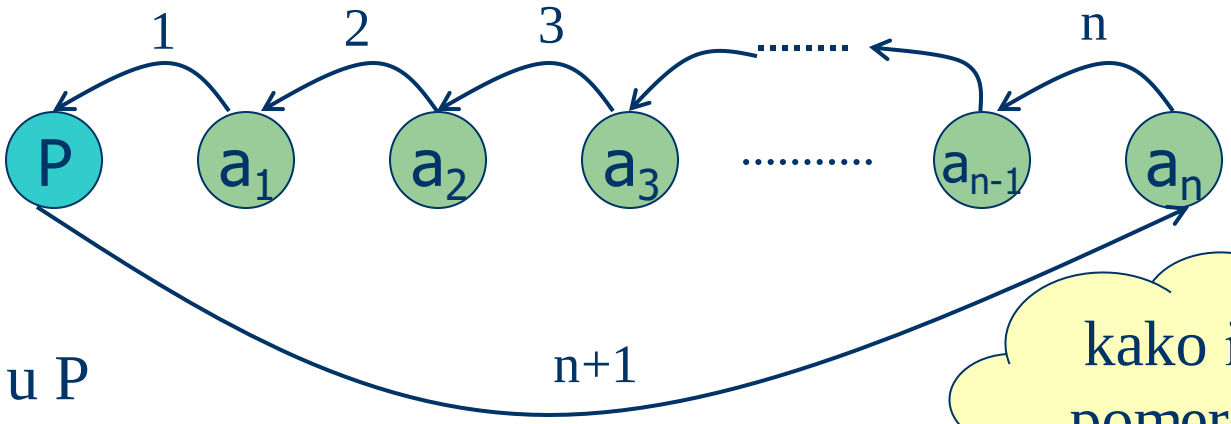
- Ciklično pomeriti u levo sadržaje lokacija A, B i C



1. iz A u P
2. iz B u A
3. iz C u B
4. iz P u C
5. kraj

Primer 3

- Ciklično pomeriti u levo sadržaje lokacija a_1, a_2, ..., a_n



- iz a_1 u P
- iz a_i u a_{i-1}, za i=2,...,n
- iz P u a_n
- kraj

kako izvršiti pomeranje za k mesta?

Euklidov algoritam za nalaženje NZD dva prirodna broja, m i n

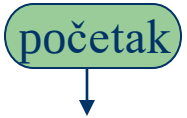
1. Podeliti m sa n i ostatak zapamtiti u r;
2. Ako je r jednako 0, NZD je n i kraj, inače preći na korak 3;
3. Zameniti m sa n, n sa r, i preći na korak 1;

Grafičko predstavljanje algoritma

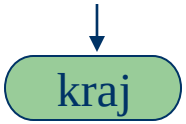
- Koriste se odredjeni grafički simboli za predstavljanje pojedinih aktivnosti u algoritmu
  - Ideja je pozajmljena iz teorije grafova
  - Algoritam se predstavlja usmerenim grafom
    - čvorovi grafa predstavljaju aktivnosti koje se obavljaju u algoritmu
    - potezi ukazuju na sledeću aktivnost koja treba da se obavi

Grafičko predstavljanje algoritma

- Polazni čvor u usmerenom grafu koji predstavlja algoritam nema dolaznih potega, a ima samo jedan izlazni poteg (granu)

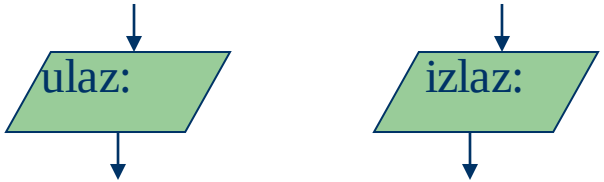


- Krajnji čvor u grafu koji predstavlja algoritam nema izlaznih potega, a ima samo jedan dolazni poteg

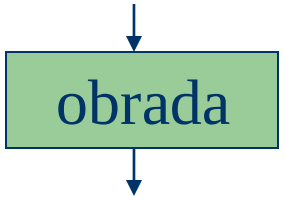


Grafičko predstavljanje algoritma

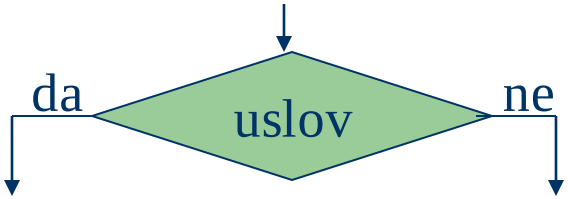
- Blok oblika romboida koristi se za označavanje ulaznih i izlaznih aktivnosti



- Blok obrade je pravougaonog oblika

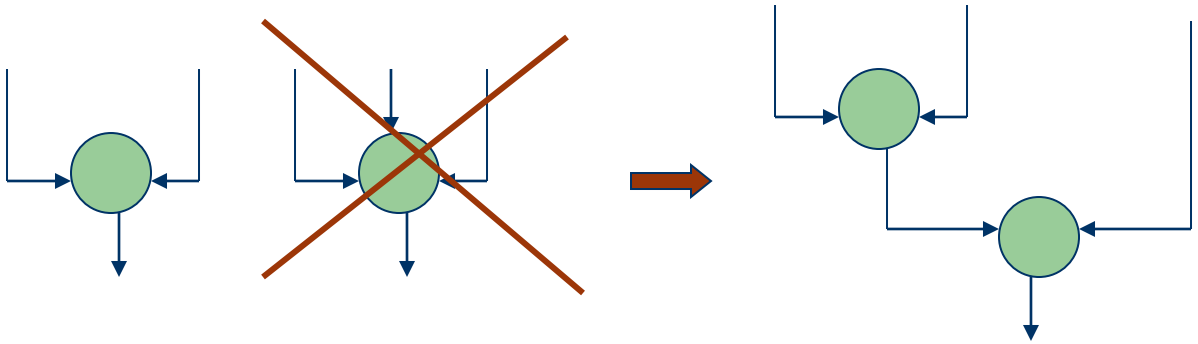


- Blok odluke



Grafičko predstavljanje algoritma

- Blok spajanja potega

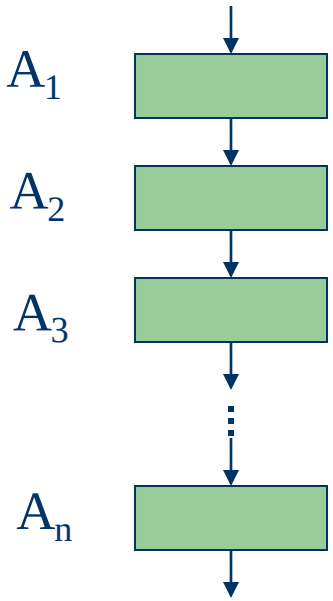


Osnovne algoritamske strukture

- Kombinovanjem gradivnih blokova dobijaju se osnovne algoritamske strukture
  - sekvenca
  - alternacija (selekcija)
  - petlja (ciklus)
- Pomoću osnovnih algoritamskih struktura može se predstaviti svaki algoritam

Sekvenca

- linijska struktura koja se dobija kaskadnim povezivanjem blokova obrade



- Algoritamski koraci se izvršavaju redom, jedan za drugim
- Algoritamski korak A_i, i=2,...,n ne može da otpočne sa izvršenjem dok se korak A_{i-1} ne završi
- sekvenca predstavlja niz naredbi dodeljivanja (:=)
- oblik naredbe:

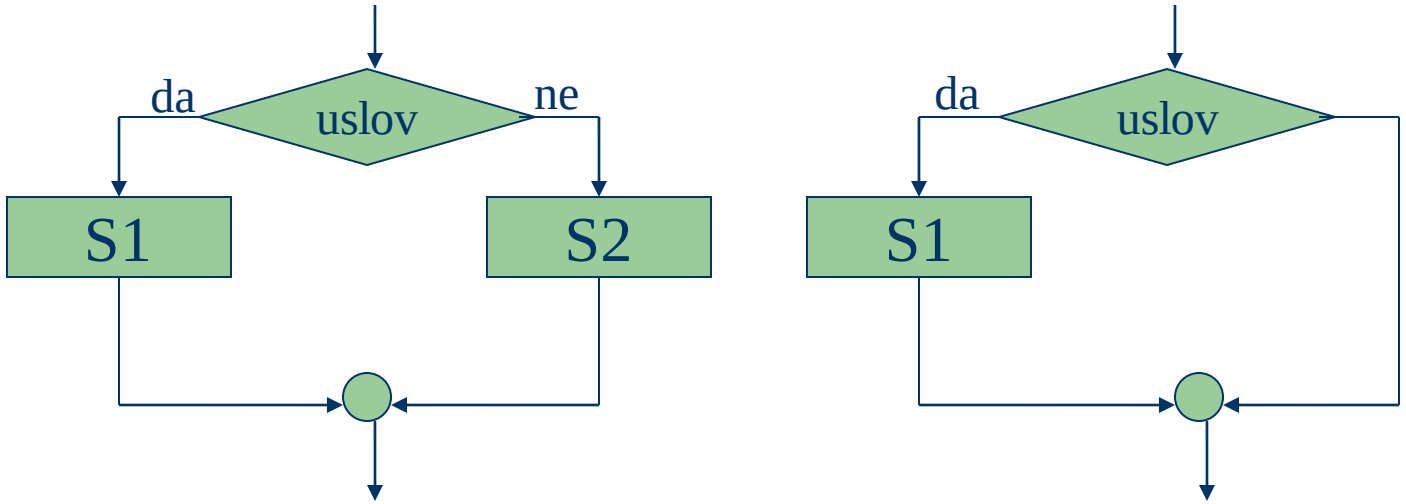
  promenljiva:=*vrednost*

  a:=b

  n:=n+1

Alternacija (selekcija)

- Omogućava uslovno izvršenje niza algoritamskih koraka

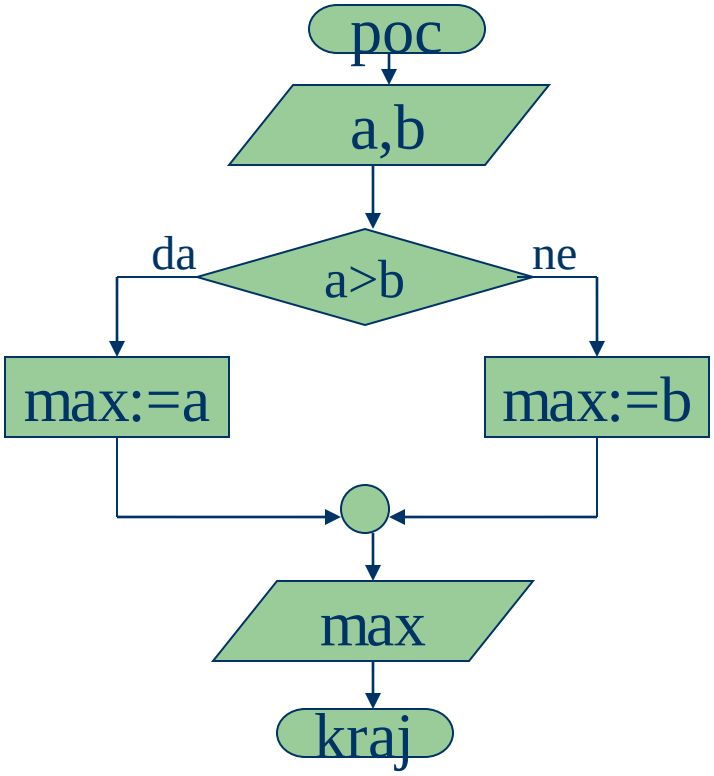


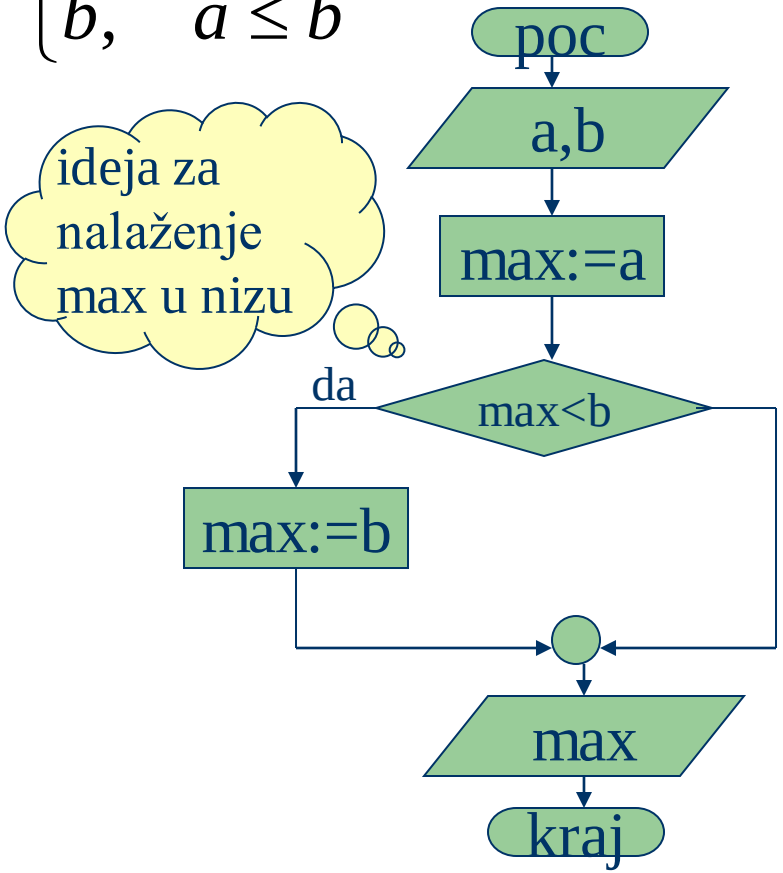
• Blokovi označeni sa S1 i S2 mogu sadržati bilo koju kombinaciju osnovnih algoritamskih struktura.

Primer 1

- Nacrtati dijagram toka algoritama kojim se odredjuje veći od dva zadata broja korišćenjem formule

$$\max\{a,b\} = \begin{cases} a, & a > b \\ b, & a \le b \end{cases}$$

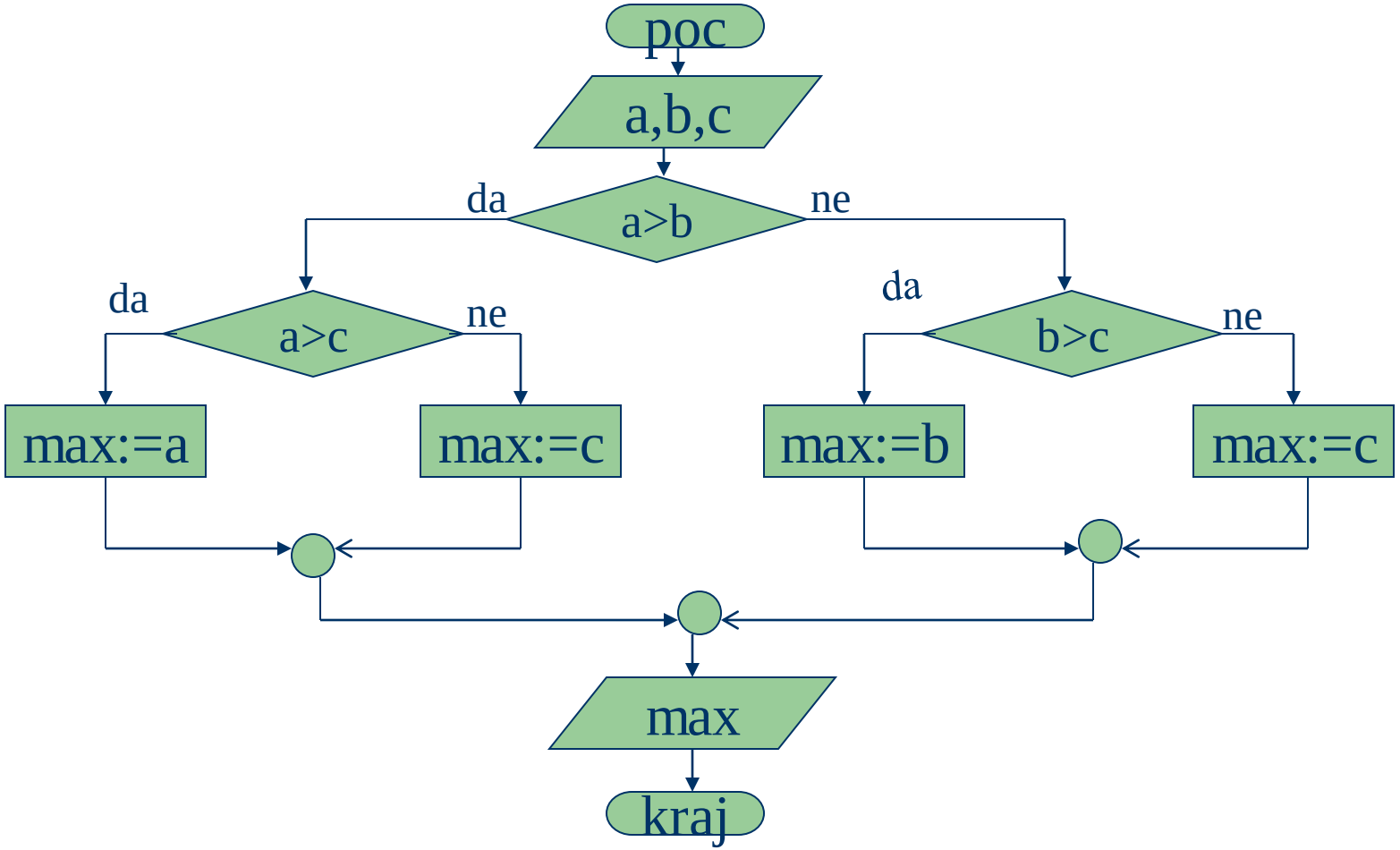




Primer 2

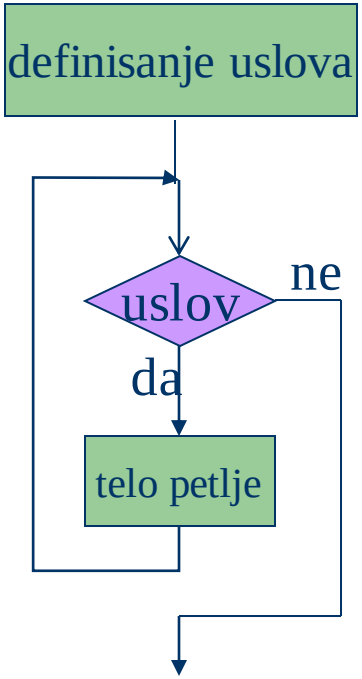
Naći maksimum od tri zadata broja a, b i c

$$\max\{a,b,c\} = \max\{\max\{a,b\},c\}$$

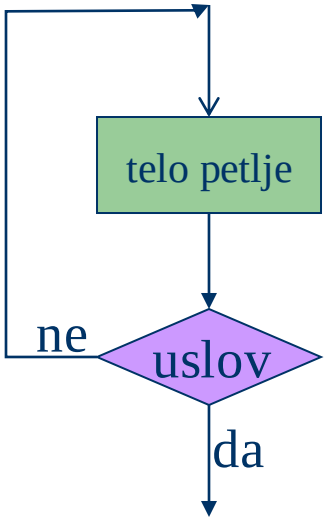


Petlja (ciklus)

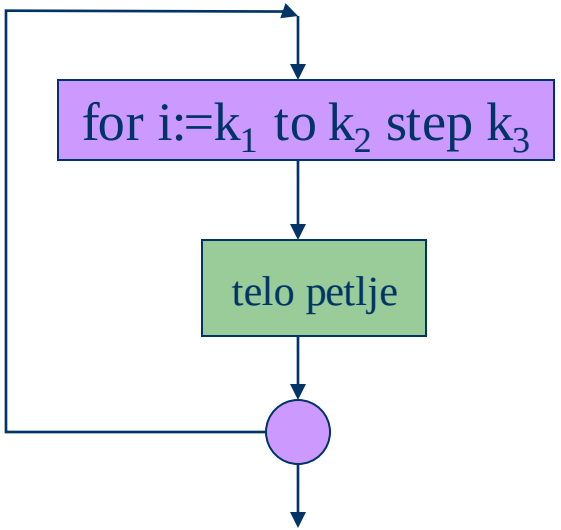
- Omogućava da se algoritamski koraci ponavljaju više puta.



while-do



repeat-until



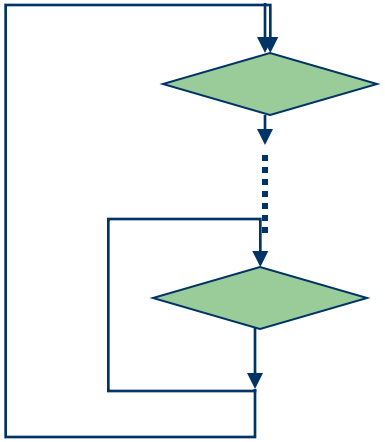
brojačka petlja

broj prolaza $n = \left[\frac{k_2 - k_1}{k_3} \right] + 1$

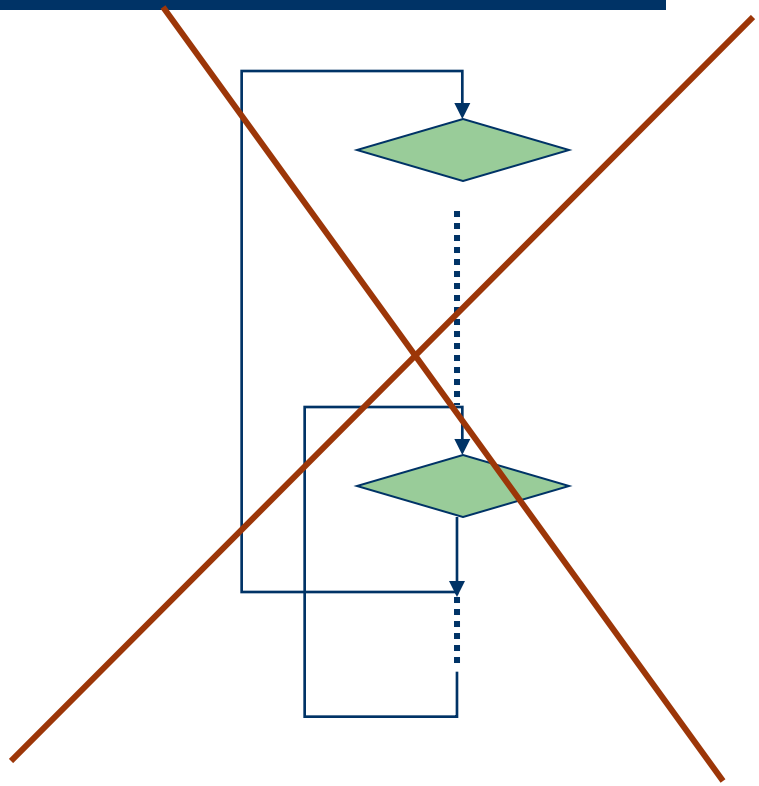
Pravila

- Ako petlja počne unutar *then* bloka ili *else* bloka, u tom bloku se mora i završiti!
- Dozvoljene su paralelne (ugnježdene) petlje.
- Nisu dozvoljene petlje koje se seku!

Pravila

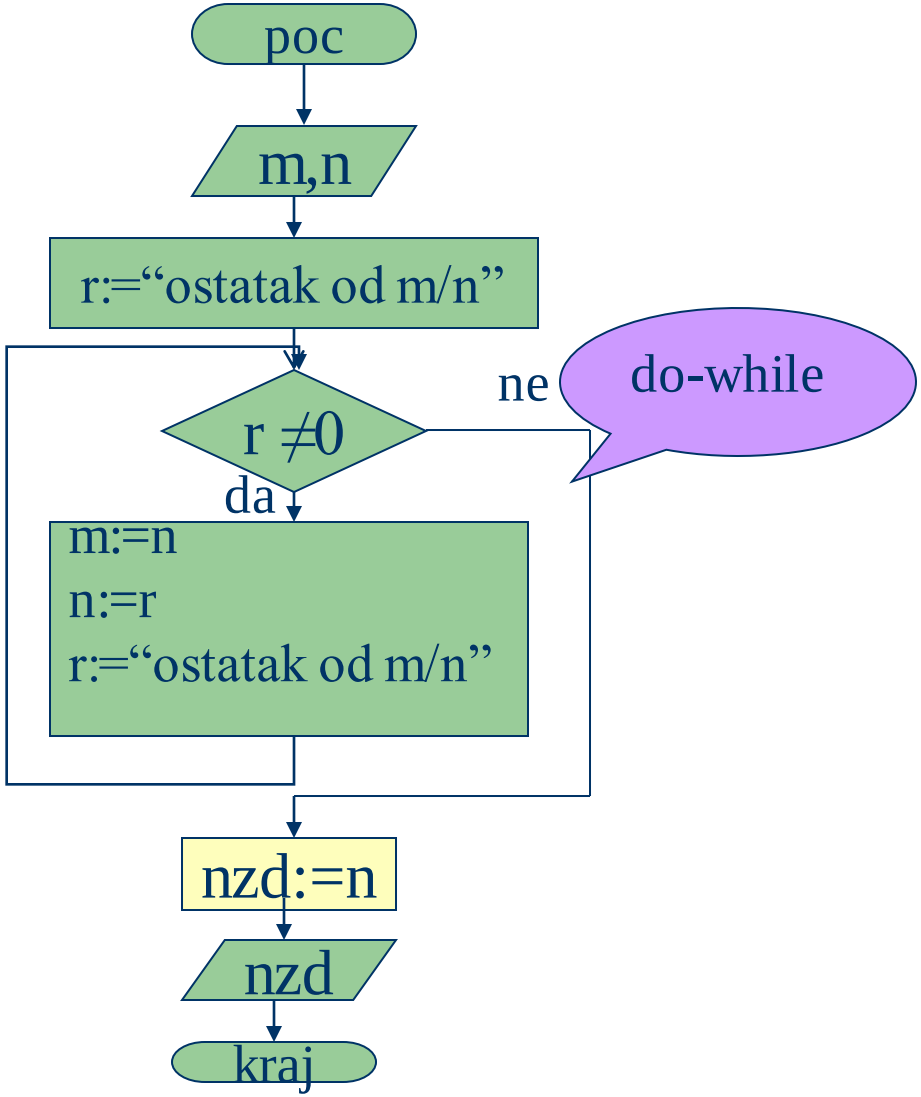


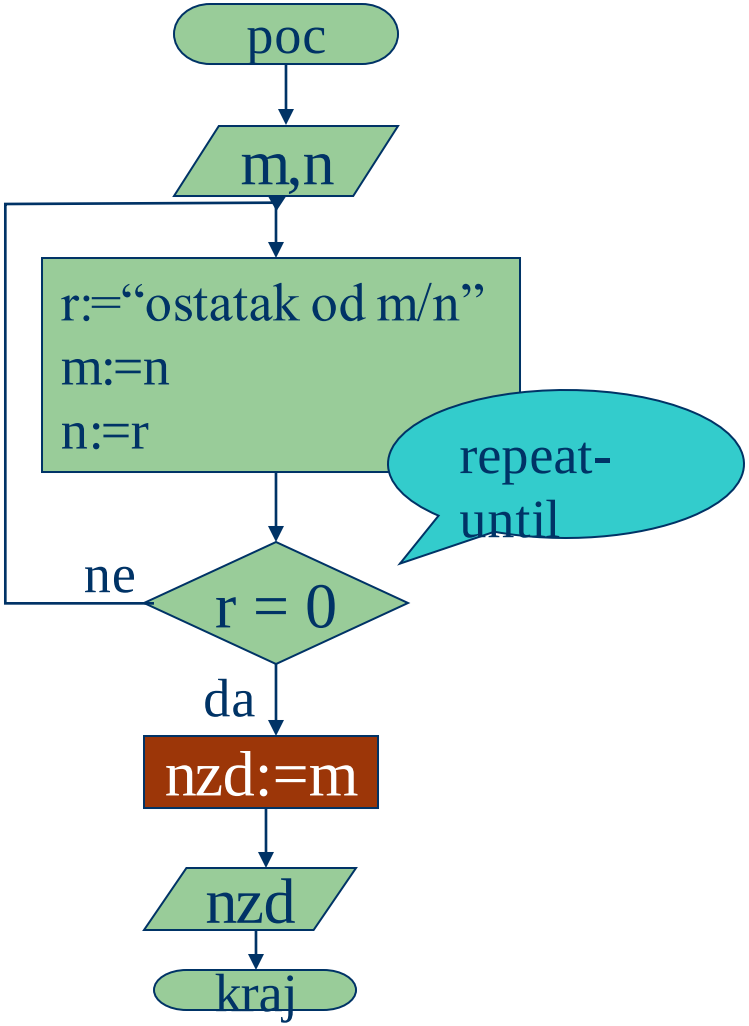
Paralelne petlje



petlje koje se seku

Primer - Euklidov algoritam





Predstavljanje algoritma pomoću pseudo koda

- Koristi se tekstualni način dopunjen formalizmom
  - svaka od osnovnih algoritamskih struktura se predstavlja na tačno definisani način:
    - **sekvenca** se predstavlja kao niz naredbi dodeljivanja odvojenih simbolom **;**

```
a:=5;
b:=a*b;
c:=b-a;
```

Predstavljanje algoritma pomoću pseudo koda

- Alternacija

```
if (uslov) then
   niz_naredbi
else
   niz_naredbi
endif;
```

ili

```
if (uslov) then
  niz_naredbi
endif;
```

```
if (a>b) then
   max:=a
else
  max:=b
endif;
```

ili

```
max:=a;
if (max<a) then
max:=b
endif;
```

Alternacija, primer 2

- max(a,b,c)

```
if (a>b) then
  if (a>c) then
     max:=a
  else
     max:=c
  endif;
else
  if (b>c) then
     max:=b
  else
     max:=c
  endif;
endif;
```

Petlje - pseudo kod

```
while (uslov) do
    niz_ naredbi
enddo;
```

Primer:

```
r:="ostatak od m/n"
while (r≠0) do
    m:=n;
    n:=r;
    r:="ostatak od m/n";
enddo;
nzd:=n;
```

```
repeat
    niz_naredbi
until (uslov);
```

Primer:

```
repeat
    r:="ostatak od m/n";
    m=n;
    n:=r;
until (r=0);
nzd:=m;
```

Strukturogrami

- Kombinacija grafičkog i pseudo koda;
  - Koriste se kao prikladna dokumentacija za već završene programe.
  - Program se piše tako da se popunjavaju odredjene geometrijske slike

Strukturogrami - primer