acs
applied computer science

UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
KATEDRA ZA PRIMENJENE RAČUNARSKE NAUKE

Osnovi programiranja i programskih jezika

Zimski semestar 2024/2025.

Studijski program: Informacioni inženjering

O predmetu

Predavači

Nastavnici

prof. dr Dinu Dragan

prof. dr Dušan Gajić

E-mail: dinud@uns.ac.rs, dusan.gajic@uns.ac.rs

Kancelarija: NTP-330

Konsultacije: dogovor putem mejla

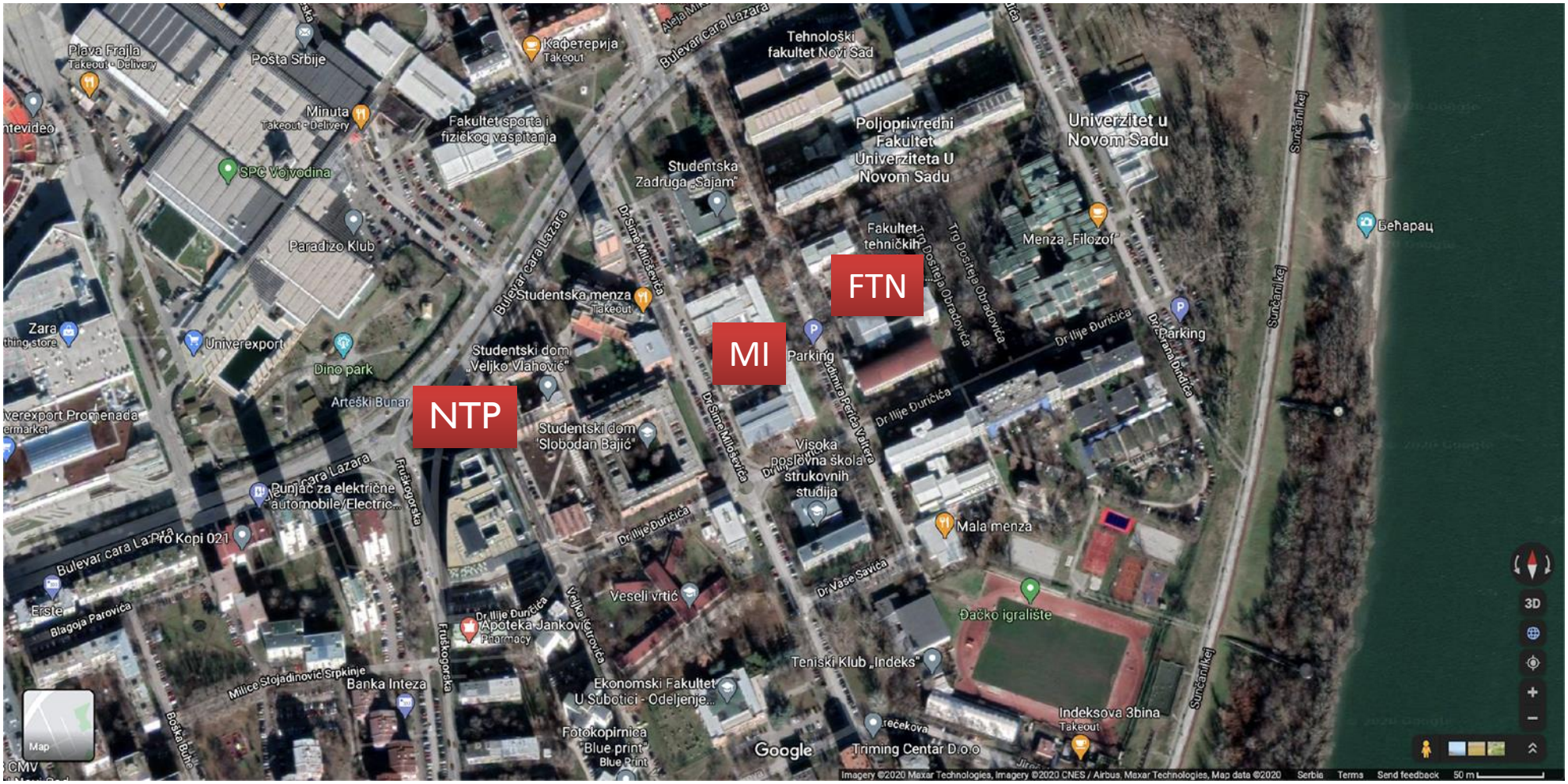
Asistenti:

Radovan Turović, radovan.turovic@uns.ac.rs

Sara Poparić, poparic.sara@uns.ac.rs

Branislav Ristić, branislav.ristic@uns.ac.rs

Zgrade FTN, MI i NTP



Teme

1. Rešavanje problema primenom računara
2. Algoritmi
   - Pojam algoritma, primeri, osobine, predstavljanje i složenost algoritama, formalizacija algoritama, Tjuringova mašina
3. Uvod u programiranje i programski jezik C
   - Formalni opis sintakse programskih jezika, programski jezik C, tipovi, operatori i izrazi, kontrola toka, potprogrami, datoteke
4. Strukture podataka, apstraktni tipovi podataka
   - Linearne strukture podataka
     - Niz, spregnuta (povezana) lista, stek, red, heš mapa
   - Nelinearne strukture podataka
     - Stablo, graf
5. Strukturirano programiranje, testiranje programa

Cilj predmeta

Ovladavanje principima i tehnikama **algoritamskog načina razmišljanja** i **pisanja programa u jeziku C**, uz poseban akcenat na **strukturama podataka**.

Organizacija ispita

Nastava: 4+4 (predavanja i računarske vežbe)

Polaganje: **predispitne** (**70%**) i **ispitne** (**30%**) **obaveze**

1. **Predispitne obaveze – ukupno 70 bodova, minimum 36**
    1. Zadatak **Z1** – 20 bodova
    2. Zadatak **Z2** – 20 bodova
    3. Zadatak **Z3** – 30 bodova

2. **Ispit – ukupno 30 bodova, minimum 16**
    1. Parcijalni ispiti u toku semestra (do 30):
        1. Parcijalni ispit (test **T1**) – do 14 bodova
        2. Parcijalni ispit (test **T2**) – do 16 bodova

Ocenjivanje

51 – 60 : 6

61 – 70 : 7

71 – 80 : 8

81 – 90 : 9

91 – 100 : 10

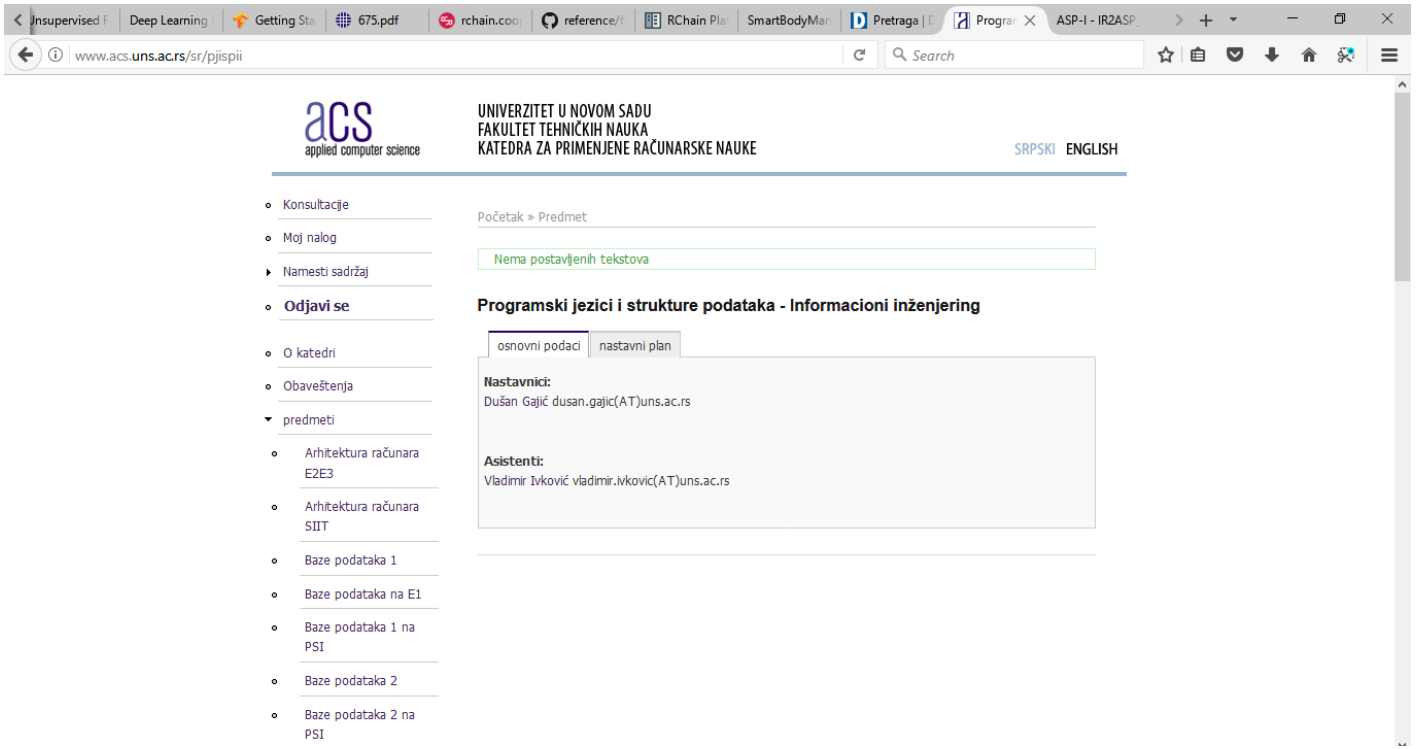
Kreativnost i priprema ispita

Sajt predmeta

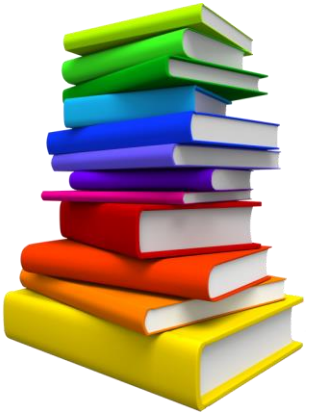
Sajt predmeta: *www.acs.uns.ac.rs*

Osnovi programiranja i programskih jezika

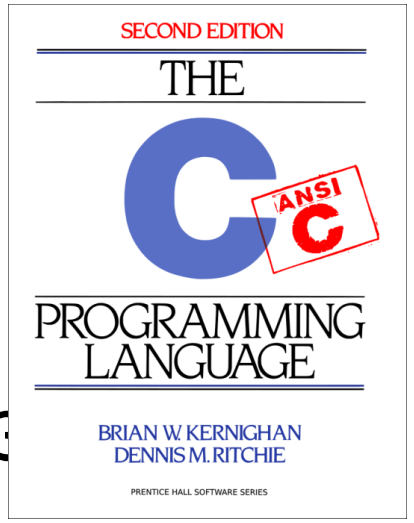
Stranica sa obaveštenjima, repozitorijum



Literatura



1. Slajdovi, prateći kod + vaše beleške
2. Brian W. Kernighan, Dennis M. Ritchie
   The C Programming Language, (*K&R*)
   2nd ed., Prentice Hall,1988.
   Srpsko izdanje:
   Programski jezik C, CET Beograd, 200



3. Laslo Kraus, *Programski jezik C sa rešenim zadacima*, 2014.
4. Milo Tomašević, *Algoritmi i strukture podataka*, 2008.
5. Dejan Živković, *Uvod u algoritme i strukture podataka*, 2010. http://www.vps.ns.ac.rs/Materijal/mat21058.pdf
6. Vladimir Ćirić, *Uvod u programiranje i programski jezik C*, 2014.

Rešavanje problema primenom računara

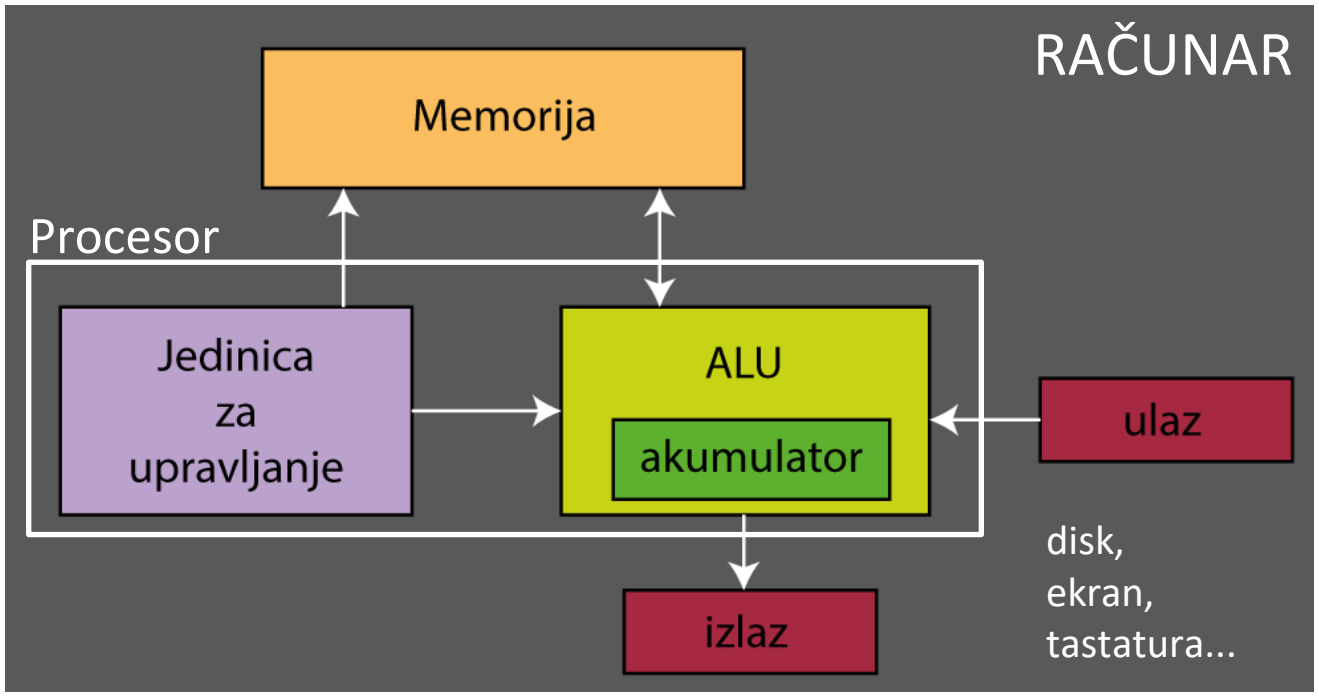
Algoritam

Algoritam je precizno definisan postupak za rešavanje nekog problema.

Računarstvo je nauka o algoritmima:

- njihovim **formalnim osobinama**
  - tačnost, ograničenja, efikasnost, cena
- njihovim **hardverskim realizacijama**
  - projektovanje računarskih sistema
- njihovim **jezičkim realizacijama**
  - programiranje i programski jezici
- njihovim **primenama**
  - inženjerstvo, finansije, medicina, fizika, biologija, hemija...

Organizacija računara



Brzi procesor/memorija naspram sporih izlazno/ulaznih uređaja

Ključni problem je savladavanje kompleksnosti

Programiranje == davanje uputstva računaru
tj. opis algoritma na jeziku razumljivom računaru

Generacije programskih jezika

1. **generacija (JEZICI VEOMA NISKOG NIVOA) – Mašinski jezici**
   - programiranje korišćenjem binarnog koda (0 i 1)
   - uporedo sa prvim komercijalnim računarima
   - komande zavise od arhitekture procesora, nema prenosa koda
   - potrebna ekspertiza u elektronici i digitalnim sistemima
   - svi programski jezici se na kraju prevode na mašinski

2. **generacija (JEZICI NISKOG NIVOA) – Asemblerski jezici**
   - simboli koji odgovaraju instrukcijama mašinskog jezika
   - prvi put se javlja kompajliranje (prevođenje na mašinski jezik)
   - manja zavisnost od hardverske platforme
   - danas se koriste za pisanje pojedinih delova sistemskog softvera i delova programa kod kojih su kritične performanse

Generacije programskih jezika

3. **generacija (JEZICI VISOKOG NIVOA) – Proceduralni i objektno-orijentisani jezici**
   - nezavisni od fizičke arhitekture hardvera
   - jedna naredba prevodi se u više naredbi mašinskog jezika
   - jezičke strukture – utvrđena struktura programskog koda
   - kompajlerski (C/C++, Java, C#, Fortran, Pascal) ili interpreterski (Python, PHP, JavaScript...) jezici
   - proceduralni (C, Pascal, Fortran, Cobol, Basic...) i objektno-orijentisani (C++, C#, Java...)

Generacije programskih jezika

4. **generacija (JEZICI VEOMA VISOKOG NIVOA) – Deklarativni i namenski jezici**
   - veoma visok nivo apstrakcije
   - namenski jezici – engl. *domain-specific languages* (DSLs)
   - zahtevaju minimalno programersko znanje, ali im je primena vrlo specifična
   - deklarativni jezici – opisuje se **šta** program treba da uradi, ali ne i **kako**
   - jezici veštačke inteligencije (Prolog)
   - jezici veoma visokog nivoa za rad sa bazama podataka (SQL)

Paradigme programskih jezika

Pogled koji programer ima ka programu i njegovom izvršenju

Neke od najznačajnijih paradigmi su:

1. **imperativna paradigma**
   - **program = algoritam + podaci (C/C++, Fortran, Pascal, Java, Go)**
   - opisuju šta treba uraditi, važan koncept stanja sistema
2. funkcionalna paradigma
   - program = funkcije višeg reda (engl. *higher-order functions*) – funkcija čiji je argument funkcija (Lisp, Scala, F#)
   - deklarativni jezici, rekurzivne funkcije i vrednosti
3. logička paradigma
   - program = činjenice + pravila (Prolog)
   - primena matematičke logike – automatsko zaključivanje na osnovu činjenica
4. objektno-orijentisana paradigma
   - program = objekti + poruke (SmallTalk, C++, Java, Python, C#)
   - programski objekti kao apstrakcije realnih objekata

Razvoj programa

"*Postoje dva načina za projektovanje i razvoj programa. Jedan je da ih učinite tako jednostavnim da je očigledno da nemaju nedostatke. Drugi je da ih učinite tako komplikovanim da nemaju očiglednih nedostataka.*"

C.A.R. Hoare

Razvoj programa jednostavne funkcionalnosti

1. Precizan opis problema

- Šta program treba da radi? Odgovor pruža **specifikacija**
- **IDENTIFIKOVATI PROBLEM** – koncizno opisati problem, preduslove za rad programa i očekivanja od rada programa
- **ODREDITI ULAZE I IZLAZE** – koje podatke program treba da dobije da bi obavio željenu radnju i koje podatke će program vratiti korisniku

2. Definisanje modela i izbor metode

- opis problema formalnim matematičkim aparatom
- tačne i približne metode, ocena složenosti izabranih modela i metoda
- **DEKOMPOZICIJA** - razlaganje problema na potprobleme sve do elementarnih problema koje se znaju rešiti i implementirati
- **APSTRAKCIJA** - izbor informacija o problemu koje su relevantne i ignorisanje informacija koje su irelevatne za dati kontekst

Razvoj programa jednostavne funkcionalnosti

3. Projektovanje algoritma

- postupak (način razmišljanja) kojim se iz datih polaznih podataka, određenim konačnim nizom računskih i/ili logičkih postupaka dolazi do traženih rezultata
- osobine: diskretnost, determinisanost, efektivnost, rezultativnost, generalnost
- primeri: recept, uputstvo za sklapanje, određivanje NZD...
- **nezavisan od programskog jezika**
- piše se u skladu sa modelom definisanim u ranijim fazama
- izbor odgovarajućih **apstraktnih tipova podataka** (ATP) (matematički (logički) model – skup podataka i operacija) i **struktura podataka** (fizička realizacija – memorijska i algoritamska reprezentacija)

Razvoj programa jednostavne funkcionalnosti

4. **Pisanje programa (programiranje)**
   - pisanje programa prema algoritmu (preslikavanje 1:N)
   - **program je implementacija algoritma na računaru instrukcijama konkretnog programskog jezika**
   - kompajlerski i interpreterski jezici

5. **Testiranje programa i pisanje dokumentacije**
   - izvršavanje programa nad test podacima sa ciljem da se pronađu greške, pisanje testova radi se pre, tokom i nakon razvoja programa
   - sintaksne i semantičke greške
   - pisanje dokumentacije je neophodan korak za proizvod

Primeri rešavanja problema

Primer 1. Kalkulator za sabiranje i oduzimanje celih brojeva.

- Šta su ulazi i izlazi?
- Kako čuvati podatke?
- Postoji li potreba za dekompozicijom?

Primer 2. Sortiranje rezultata ispita prema broju poena ili grupama kojima pripadaju studenti.

- Šta su ulazi i izlazi?
- Kako čuvati podatke?
- Postoji li potreba za dekompozicijom?

Apstrakcija u razvoju programa

Dekompozicija i **apstrakcija** su dve **glavne tehnike** za **upravljanje složenošću programa**

"*Suština apstrakcije je čuvanje informacija koje su relevantne u datom kontekstu i ignorisanje informacija koje su irelevatne za dati kontekst*"

John Guttag

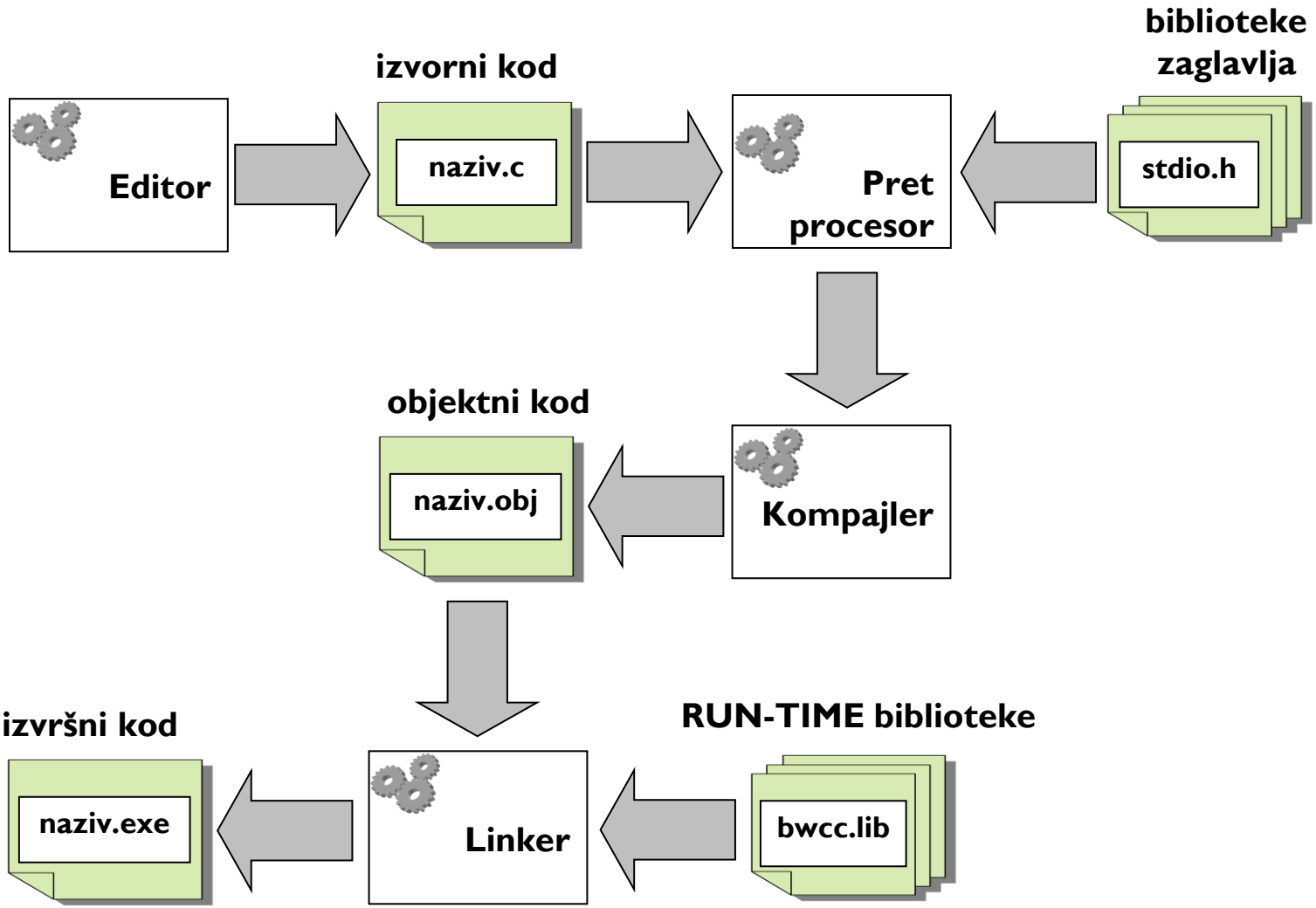
Primeri:

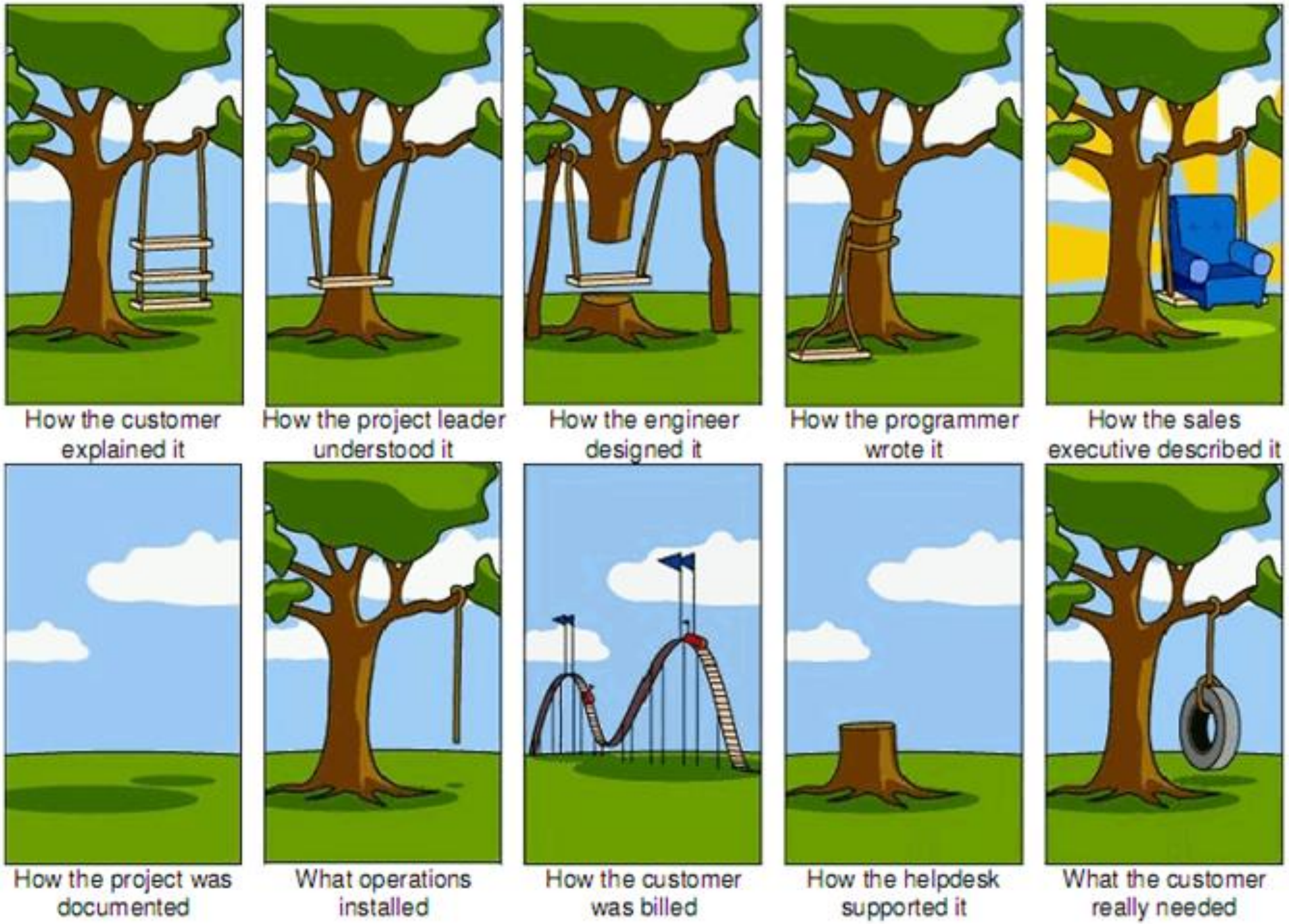
Čovek je apstrakcija studenta, lekara, vozača...

Životinja je apstrakcija konja, delfina, psa...

Živo biće je apstrakcija čoveka, životinje, biljke...

Proces prevođenja programa




How the customer explained it
How the project leader understood it
How the engineer designed it
How the programmer wrote it
How the sales executive described it
How the project was documented
What operations installed
How the customer was billed
How the helpdesk supported it
What the customer really needed

Algoritmi

Pojam algoritma

ALGORITAM = UPUTSTVO = RECEPT

Algoritam je precizno definisan postupak sa konačnom listom koraka za rešavanje nekog problema. **Algoritam** prihvata **ulazne vrednosti** i proizvodi **izlazne vrednosti.**

Termin "algoritam" predstavlja latiniziranu transkripciju imena poznatog naučnika iz IX veka iz **Muhameda Al-Horezma**, tj. Muhameda iz Horezma (grad u današnjem Uzbekistanu)

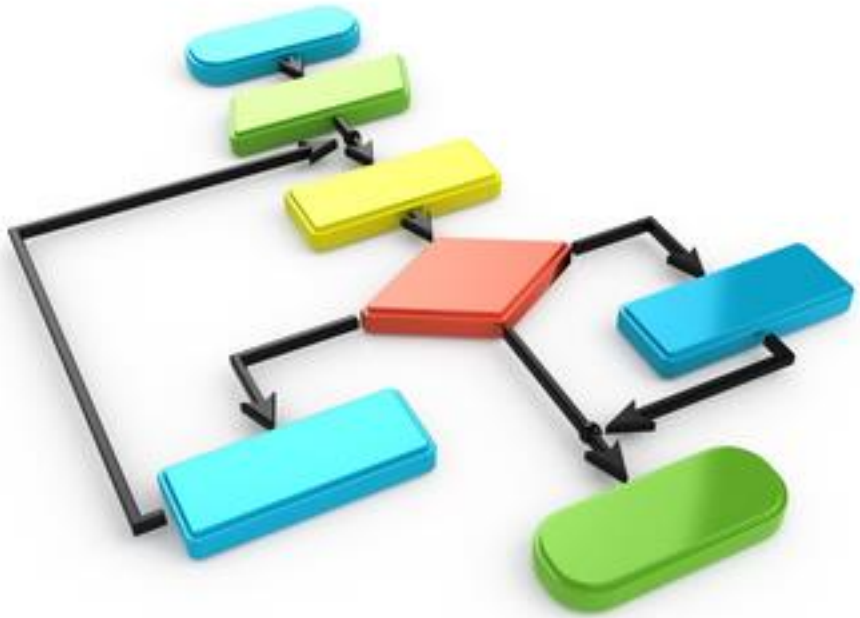
Primeri algoritama

1. Kuvanje

1. Kretanje u gradu

1. Množenje

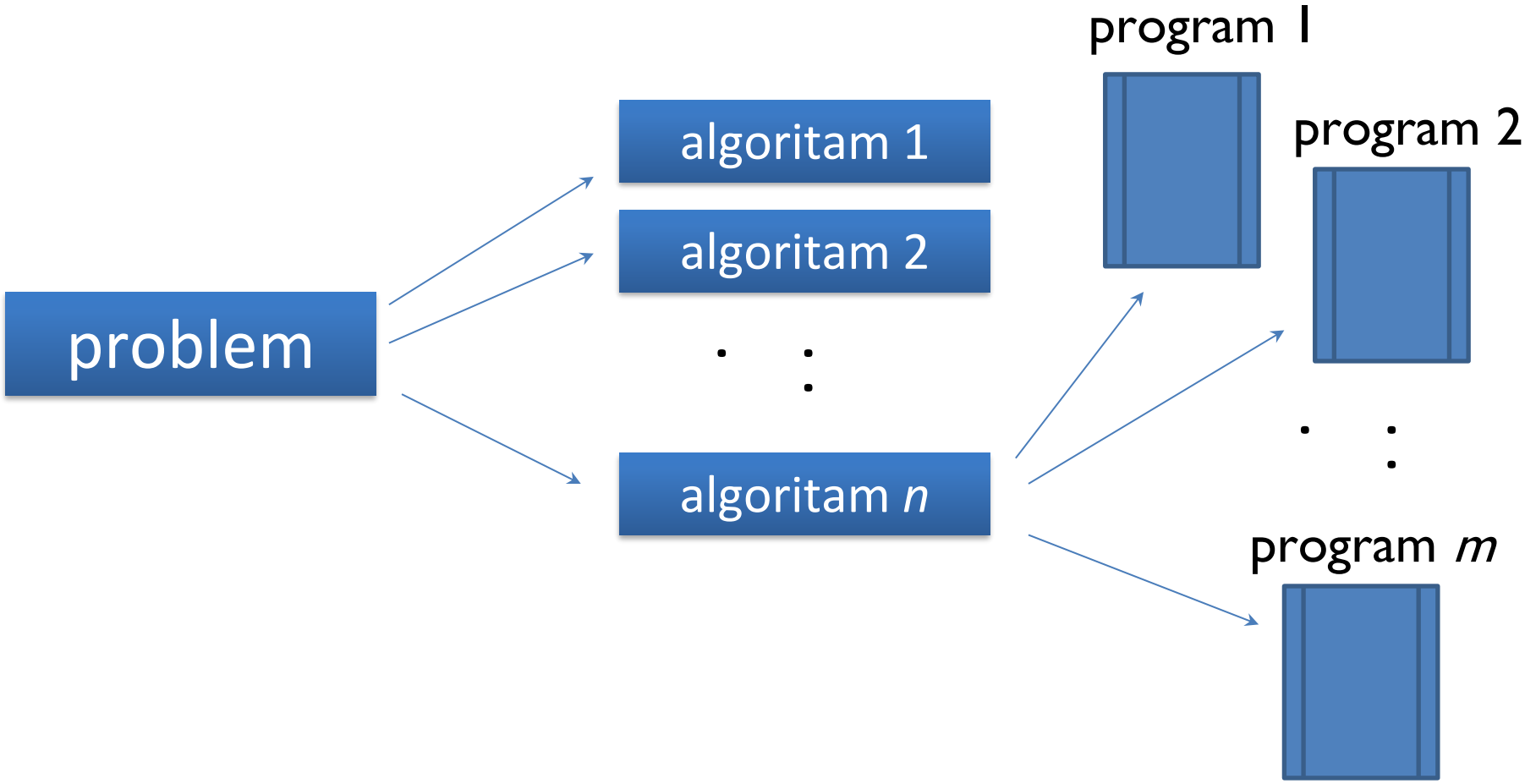
1. Određivanje NZD



Odnos problema, algoritma i programa

Za rešavanje svakog problema, mogu postojati različiti algoritmi

Za svaki algoritam, mogu postojati različite implementacije (programi)



Osobine algoritma

Podrazumeva se da algoritam mora prvo da bude **ispravan** (**tačan**)!

1. Diskretnost
2. Determinisanost
3. Efektivnost (konačnost)
4. Rezultativnost
5. Generalnost (masovnost)
6. (Optimalnost)

Diskretnost

Algoritam se sastoji od konačnog broja koraka

Svaki korak zahteva obavljanje jedne ili više operacija

U zavisnosti od operacija koje računar može da obavi, uvode se ograničenja za tip operacija koje se mogu koristiti u algoritmu

Mašina Minskog: Tjuring-kompletnost* zahteva četiri tipa operacija – uslovni skok, bezuslovni skok, dodela, zaustavljanje

* Tjuring-kompletan je svaki sistem pravila za rad sa podacima (skup instrukcija, programski jezik) kojim se može simulirati rad bilo koje Tjuringove mašine

Determinisanost

Svaki algoritamski korak mora biti precizno definisan i potpuno jasan

- Izrazi „13/0“ ili „oduzmi 4 ili 5 od trenutne vrednosti proizvoda“ nisu dozvoljeni

Posle izvršavanja tekućeg koraka u algoritmu mora biti jednoznačno određeno koji je sledeći korak

Efektivnost (konačnost)

Sve operacije koje se javljaju u algoritamskim koracima moraju se izvršiti za konačno (razumno kratko) vreme i moraju biti dovoljno jednostavne da se mogu tačno izvršiti

Vreme izvršavanja celog algoritma mora biti konačno, tj. prihvatljivog trajanja

Rezultativnost

Svaki algoritam mora posle konačnog broja koraka generisati traženi rezultat

Algoritam može imati nula ili više ulaznih podataka, a može generisati jednu ili više izlaznih vrednosti

Generalnost (masovnost)

Svaki algoritam definiše postupak za rešavanje klase problema, a ne pojedinačnog slučaja

- Pretraživanje ili sortiranje bilo kog podskupa celih ili realnih brojeva, množenje matrica/vektora bilo kog reda...

Načini predstavljanja algoritama

1. Tekstualni opis na prirodnom jeziku
2. Grafički (pomoću dijagrama toka)
3. Pseudokod
4. (Strukturogram)
5. Programski jezik

Tekstualni opis algoritama

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 same prirode jezika

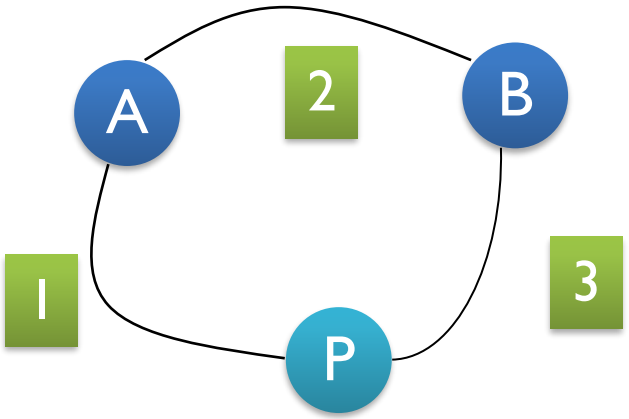
Primer 1: Euklidov algoritam za nalaženje NZD dva prirodna broja *a* i *b*

1. Podeliti *a* sa *b* i ostatak zapamtiti u *r*
2. Ako je *r* jednako 0, NZD je *b* i došli smo do kraja algoritma, inače preći na korak 3.
3. Zameniti *a* sa *b*, *b* sa *r*, i preći na korak 1

Primer 2: Zamena sadržaja dve memorijske lokacije

Zameniti sadržaje dve memorijske lokacije A i B

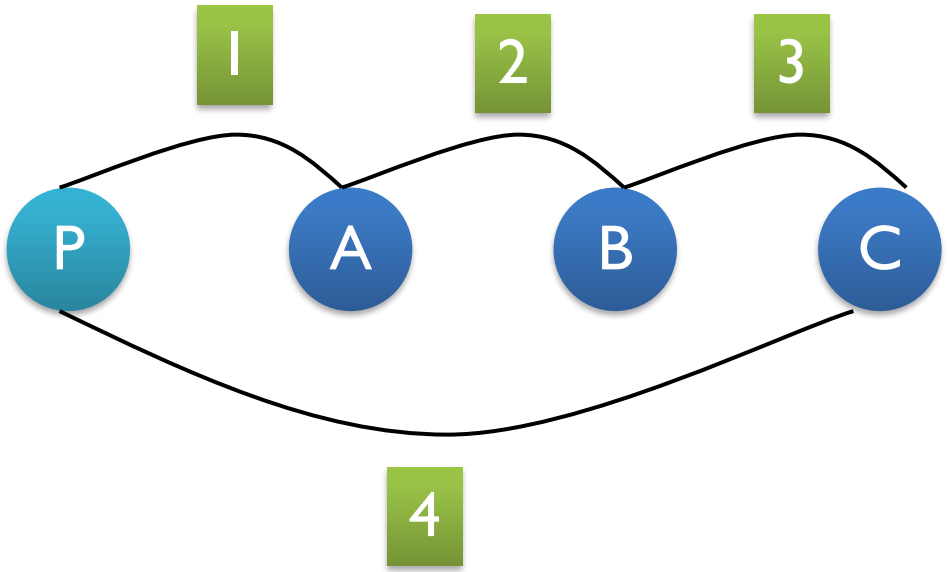
- Rešenje – potrebna je treća, pomoćna, lokacija
  1. sadržaj lokacije A zapamtiti u pomoćnoj lokaciji P
  2. sadržaj lokacije B zapamtiti u lokaciji A
  3. sadržaj lokacije P zapamtiti u lokaciji B
  4. kraj



Primer 3: Pomeranje sadržaja lokacija ulevo

Ciklički 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



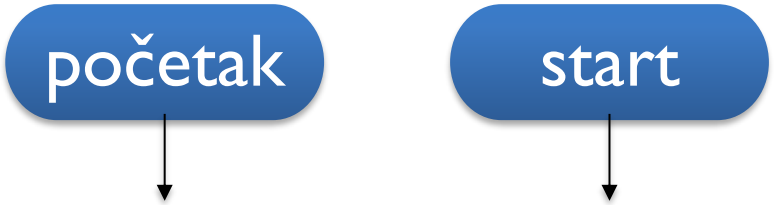
Grafičko predstavljanje algoritama

Koriste se grafički simboli za predstavljanje pojedinih aktivnosti u algoritmu

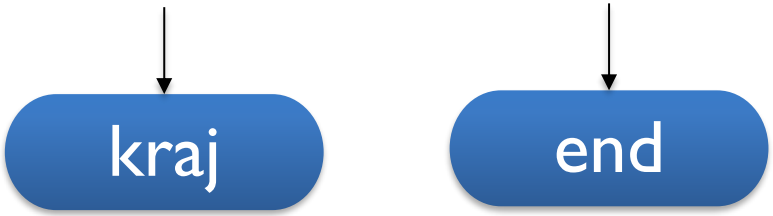
- Ideja je pozajmljena iz teorije grafova
- Algoritam se predstavlja usmerenim grafom
  - čvorovi grafa predstavljaju aktivnosti koje se obavljaju u algoritmu
  - potezi ukazuju na sledeću aktivnost koja treba da se obavi

Grafičko predstavljanje algoritama

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



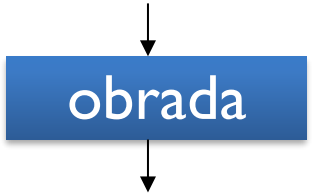
Krajnji čvor u grafu koji predstavlja algoritam nema izlaznih potega, a ima samo jedan dolazni poteg



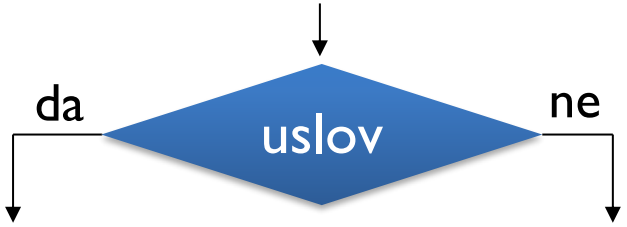
Grafičko predstavljanje algoritama

Blok oblika romboida koristi se za označavanje ulaznih i izlaznih aktivnosti

Blok obrade je pravougaonog oblika

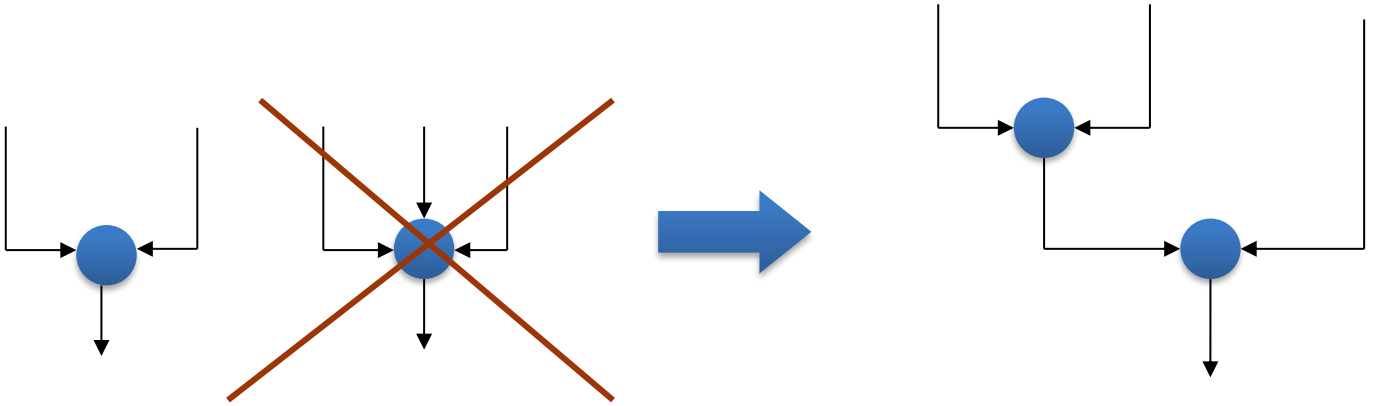


Blok odluke



Grafičko predstavljanje algoritama

Blok spajanja potega



Osnovne algoritamske strukture

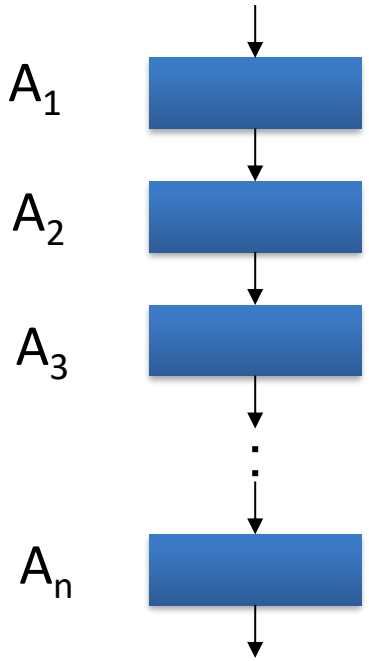
Kombinovanjem gradivnih blokova dobijaju se **osnovne algoritamske strukture**

- **sekvenca**
- **selekcija (grananje, alternacija)**
- **iteracija (petlja, ciklus)**

Pomoću **osnovnih algoritamskih struktura** može se **konstruisati dijagram toka algoritma** koji realizuje **svaku izračunljivu funkciju**
(Teorema Bohm-Jacoppini 1966.)

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:

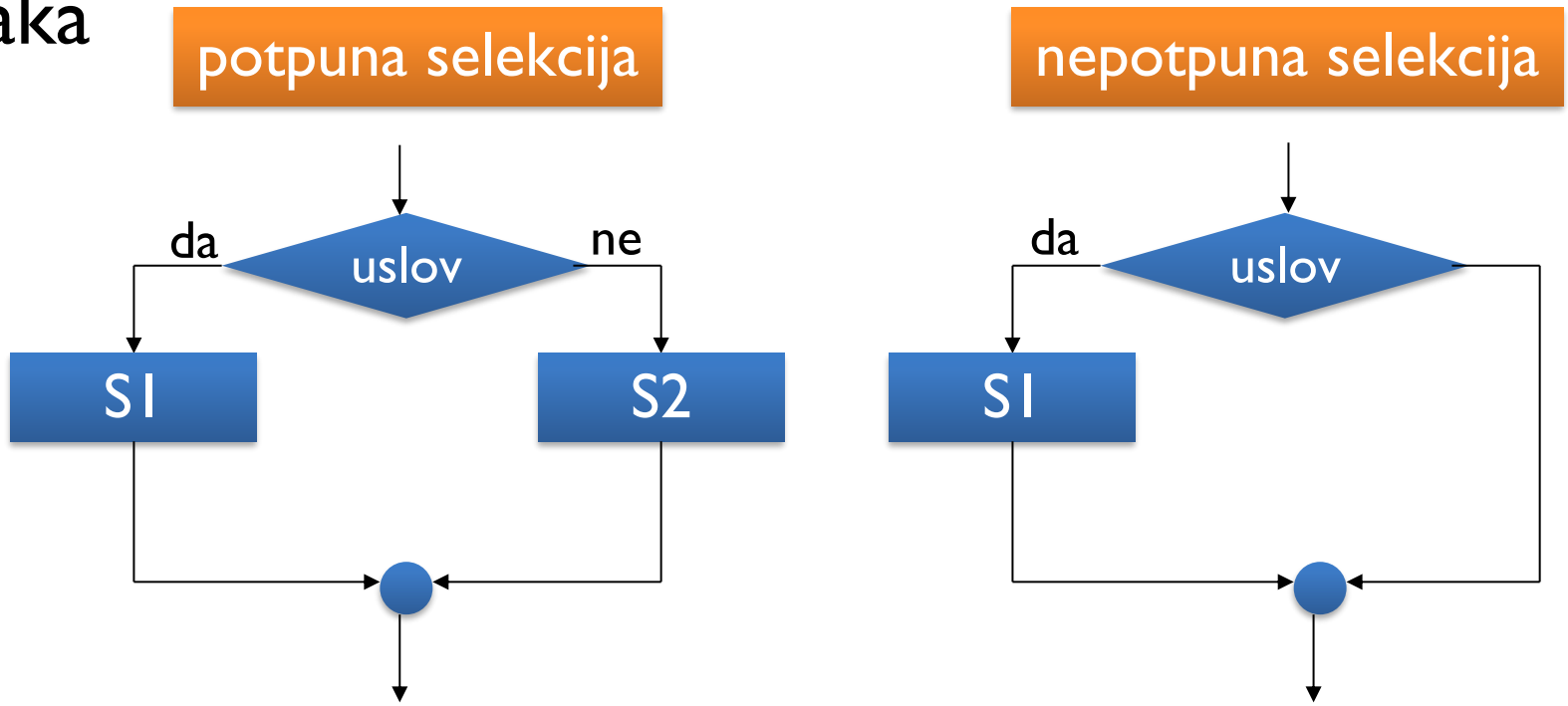
  promenljiva:=vrednost

  a:=b

  n:=n+1

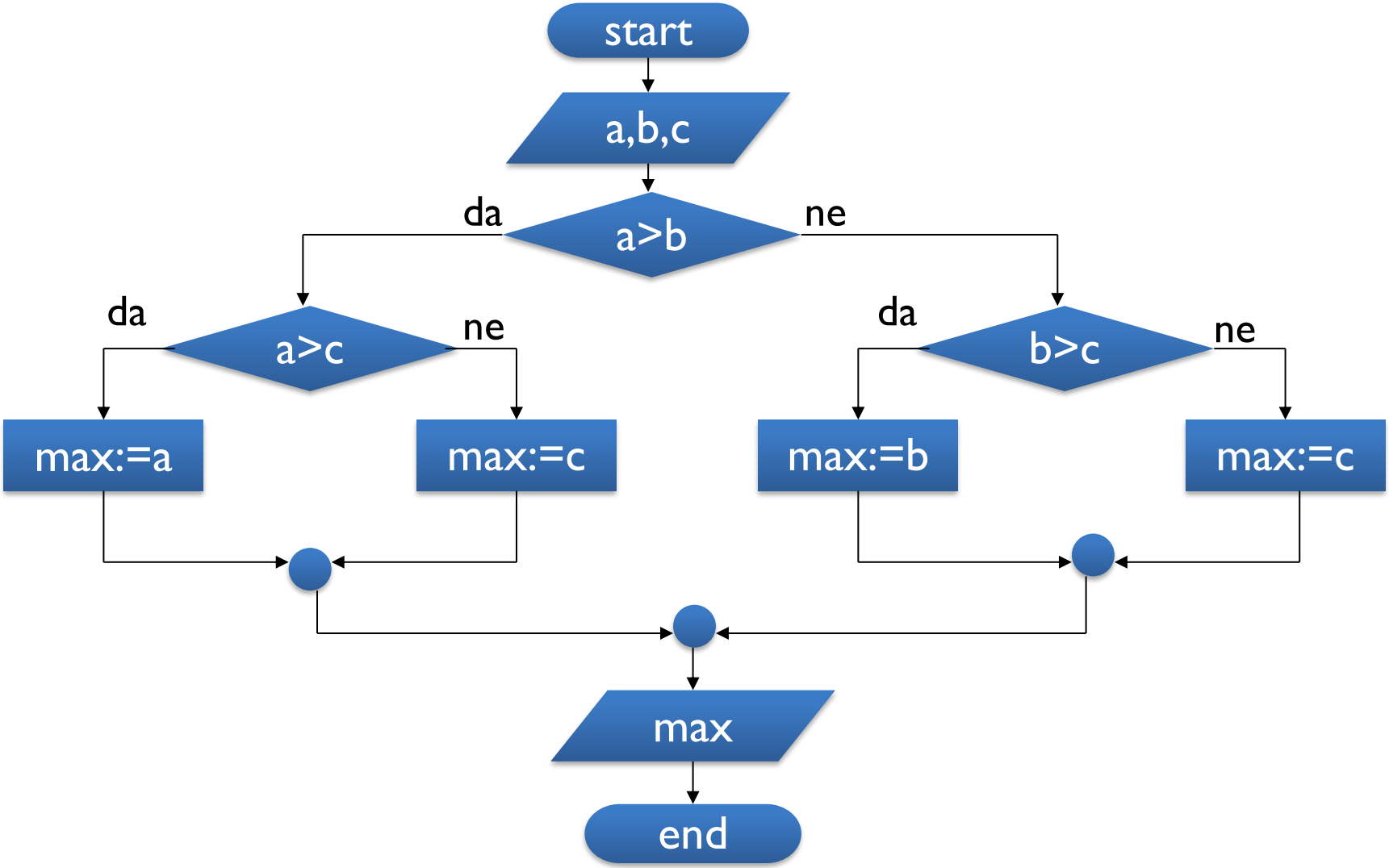
Selekcija

Omogućava uslovno izvršenje niza algoritamskih koraka



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

Primer - maksimum tri broja

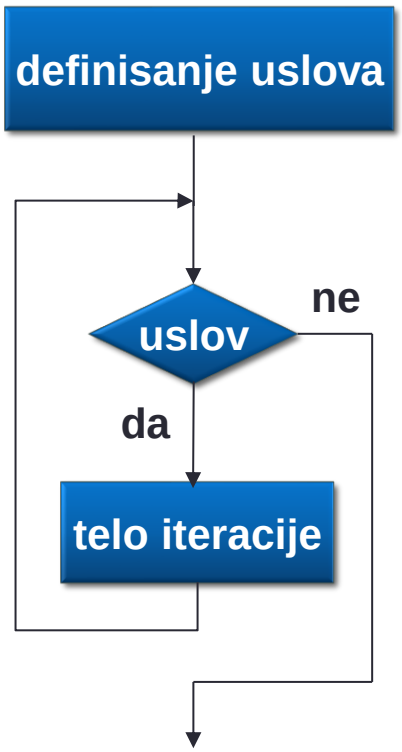


$$\max\{a,b,c\} = \max\{\max\{a,b\},c\}$$

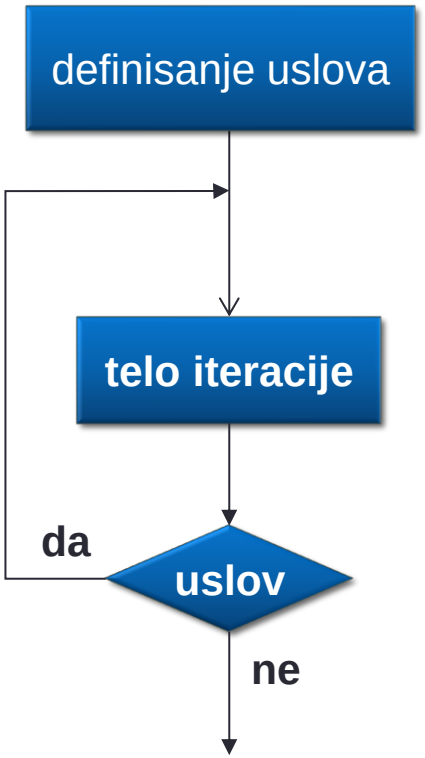
Iteracija

Omogućava da se algoritamski koraci ponavljaju više puta.

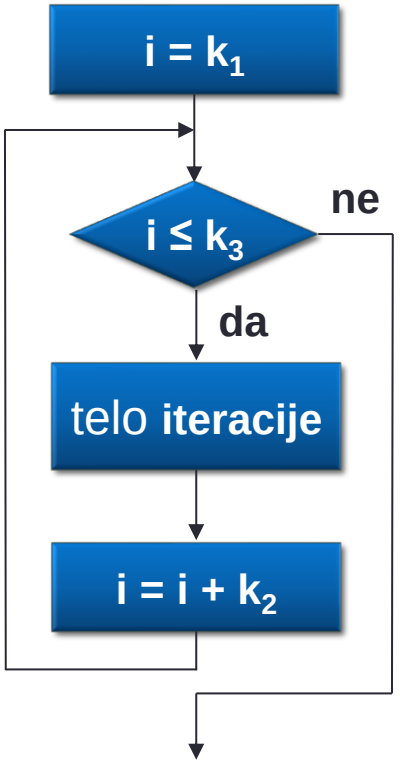
while



repeat-until



for (brojačka petlja)



broj prolaza:

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

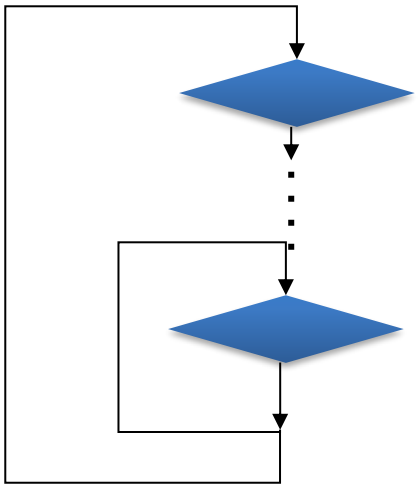
Pravila za iteracije

Ako iteracija počne unutar *then* bloka ili *else* bloka, u tom bloku se mora i završiti!

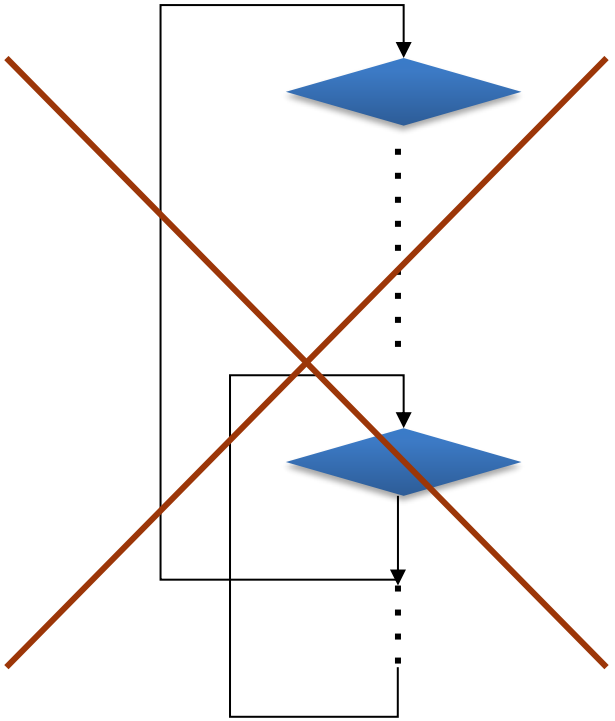
Dozvoljene su paralelne (ugnježdene) iteracije.

Nisu dozvoljene iteracije koje se seku!

Pravila za iteracije

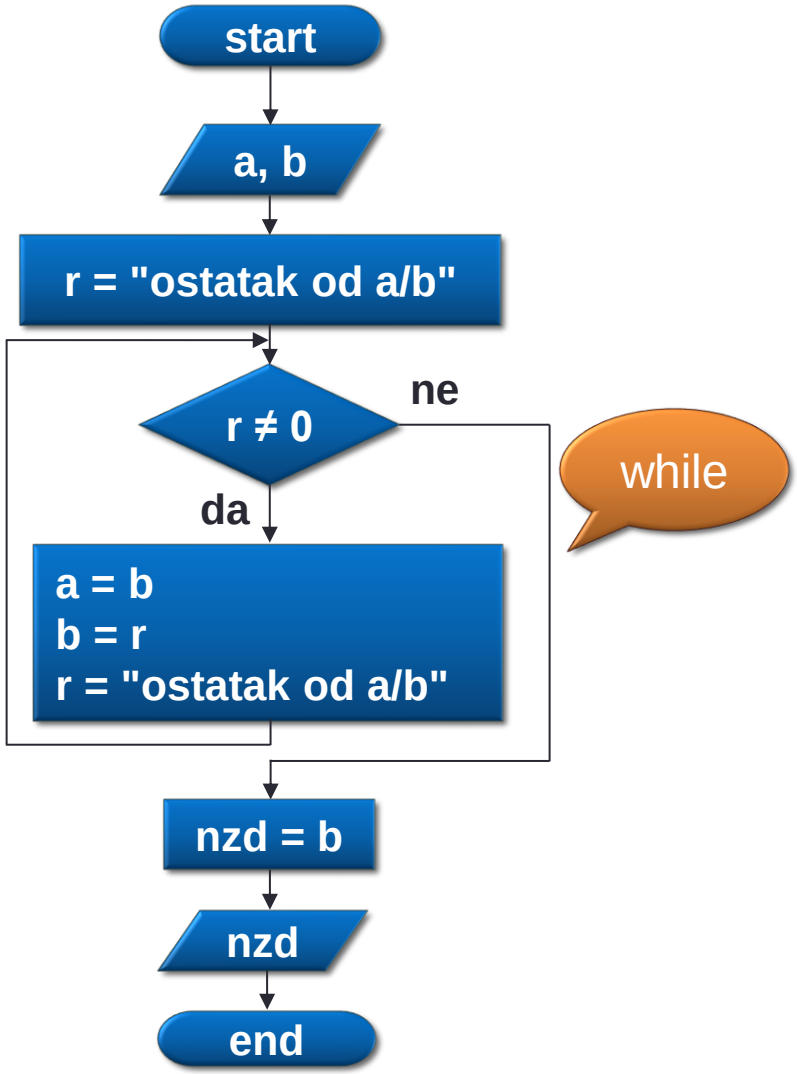


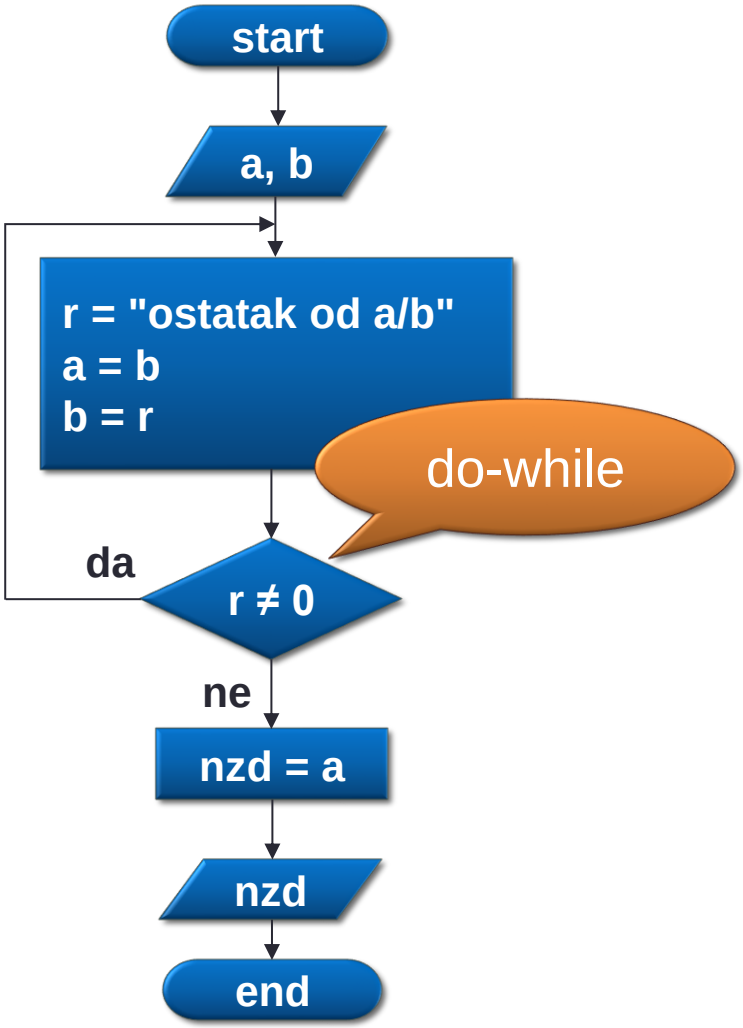
paralelne
iteracije



iteracije koje
se seku

Primer – Euklidov algoritam





Predstavljanje sekvence pseudokodom

Koristi se tekstualni način dopunjen formalizmom

Sekvenca se predstavlja kao niz naredbi dodeljivanja odvojenih simbolom **;**

```
a:=5;
b:=7;
b:=a*b;
c:=b-a;
d:=c*c;
…
```

Predstavljanje selekcije pseudokodom

Potpuna selekcija:

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

Primer:

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

ili

Nepotpuna selekcija:

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

Primer:

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

Predstavljanje iteracija pseudokodom

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

Primer:

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

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

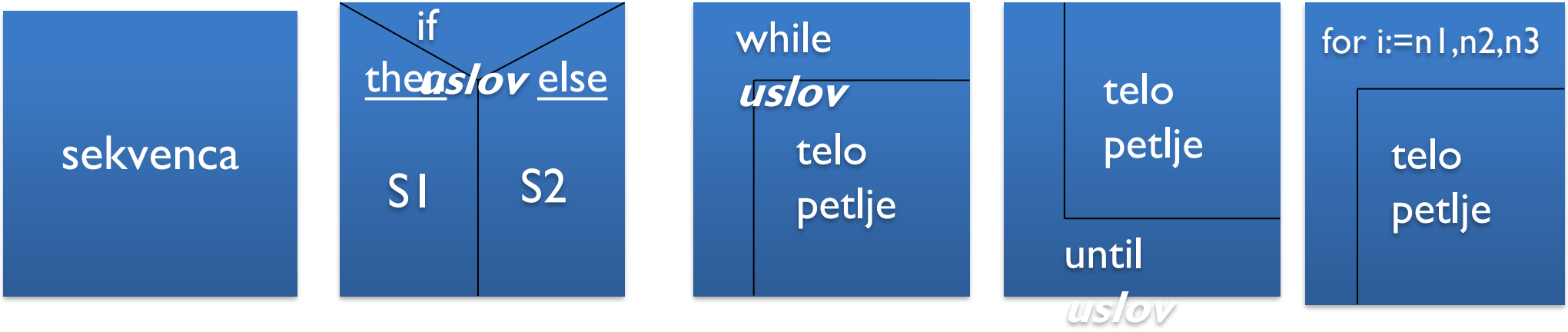
Primer:

```
repeat
    r:="ostatak od a/b";
    a:=b;
    b:=r;
until (r=0);
nzd:=a;
```

Strukturogrami

Kombinacija grafičke reprezentacije i pseudokoda

- Mogu se kreirati samo za strukturirane programe
- Koriste se kao prikladna dokumentacija za već završene programe.
- Program se piše tako da se popunjavaju odredjene geometrijske slike



Primer – maksimum tri broja