# ALGORITMI I STRUKTURE PODATAKA

- PREDAVANJE I -

Profesor: dr Slavimir Stošović, dipl. inž. el.

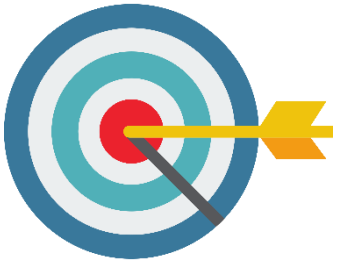
Asistent: Miloš Kosanović, dipl. inž. el.

# SADRŽAJ, CILJEVI I PREDUSLOVI

- Sadržaj:
  - Zašto algoritmi i strukture podataka?
  - Podaci i informacije
  - Metode programiranja
  - Algoritmi - metode za rešavanje problema
  - Strukture podataka – metode za čuvanje podataka
  - Implementacija algoritma u programski jezik
  - Uvod u programske jezike
  - Programska okruženja – platforme za razvoj i eksploataciju softvera

# SADRŽAJ, CILJEVI I PREDUSLOVI

- Ciljevi:

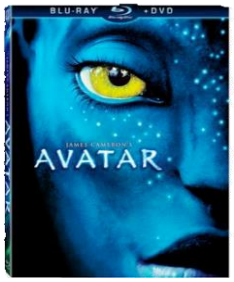


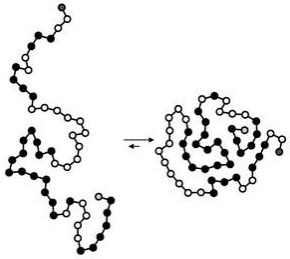
  - Edukacija studenata iz oblasti principa i teorije programiranja uz pomoć algoritamskih struktura.
  - Edukacija studenata za praktičnu primenu algoritamskih struktura i njihovo prevođenje u neki od programskih jezika.
  - Napraviti osnovu za lako savladavanje bilo kog programskog jezika.
  - Čak iako student nema nameru da postane profesionalni programer, u ovom predmetu može naći neke korisne metode i tehnike koje će poboljšati njegovu opštu sposobnost rešavanja problema.
  - Postavljanje dobre osnove za studiranje drugih predmeta iz oblasti programskih jezika, programiranja i informacionih tehnologija.
  - Uniformisati nivo znanja svih studenata prve godine!

# SADRŽAJ, CILJEVI I PREDUSLOVI

- Zašto učiti algoritme?
  - Internet – Web pretraga, rutiranje paketa, deljenje fajlova i distribucija,…
  - Saobraćaj – Red letenja na aerodromima, sortiranje prtljaga…
  - Biologija – Genetska istraživanja, čuvanje proteina,…
  - Računari – Layout čipova, fajl sistemi, kompajleri,…
  - Računarska grafika – Filmovi, video igre, virtuelna realnost,…
  - Bezbednost – Mobilni uređaji, e-commerce, mašine za glasanje,…
  - Multimedija – MP3, JPG, DivX, HDTV, prepoznavanje lica,…
  - Društvene mreže – Preporuke, news feed, reklame,…

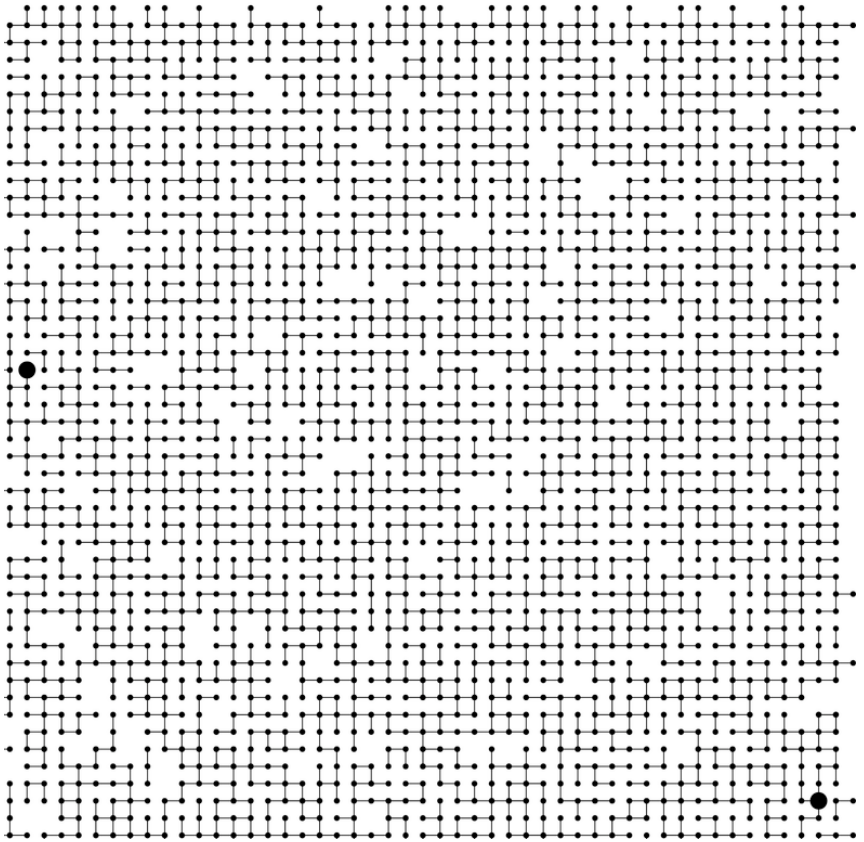
Google YAHOO! bing



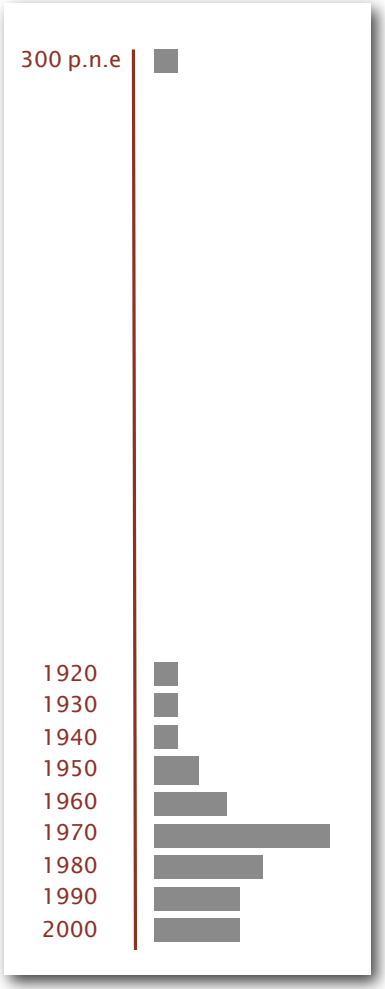
# SADRŽAJ, CILJEVI I PREDUSLOVI

- Zašto učiti algoritme?
  - Rešavanje problema koji ne mogu biti rešeni na drugi način.
  - Primer. Povezanost mreža.



# SADRŽAJ, CILJEVI I PREDUSLOVI

- Stari koreni – nove mogućnosti:
  - Učenje algoritama datira još od Euklida.
  - Formalizovano je od strane Church-a i Turing-a 1930.
  - Neki danas važni algoritmi su otkriveni na fakultetima u okviru predmeta sličnih ovom.



# SADRŽAJ, CILJEVI I PREDUSLOVI

- Zašto učiti algoritme?

> *“ Za mene, značajni algoritmi su poezija računarstva. Deluju misteriozno, ali kada se jednom otključaju oni uvode novo svetlo u oblast računarstva.“*
>
> *- Francis Sullivan (glumac)*

FROM THE EDITORS

THE JOY OF ALGORITHMS

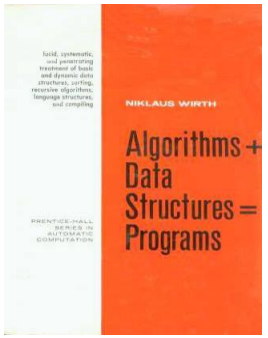
Francis Sullivan, Associate Editor-in-Chief

# SADRŽAJ, CILJEVI I PREDUSLOVI

- Zašto učiti algoritme?
  - Da možete da rešavate probleme na pravi način.
  - Da postanete vrhunski programer.

> *“ Loš programer postaje dobar programer kada počne da brine i o kodu i o strukturi podataka. Loš programer brine samo o kodu. Dobar programer brine i o strukturi podataka i o njihovim relacijama.“*
> *— Linus Torvalds (tvorac Linux-a)*

> *“ Algoritmi + Strukture podataka = Program”*
> *- Niklaus Wirth (tvorac Pascal-a)*



- Zašto učiti algoritme?
  - Za zabavu i zaradu.

# SADRŽAJ, CILJEVI I PREDUSLOVI

- Preduslovi:
  - Nije potrebno nikakvo posebno predznanje i iskustvo za uspešno savladavanje ovog predmeta.
  - Svi koncepti i termini biće objašnjeni na pogodan i jasan način.
  - Računske vežbe su sastavni deo ovog predmeta i pružiće studentima znanja i veštine za praktično savladavanje inženjerskih zadataka i problema.
  - Na taj način studenti dobijaju mogućnost za proveru i utvrđivanje usvojenih znanja.

# SADRŽAJ, CILJEVI I PREDUSLOVI

- Preduslovi:
  - Želja za unapređenje znanja u ovoj oblasti.
  - Logičko razmišljanje i dedukcija.
  - Poznavanje osnovnih algoritamskih koraka iz realnog sveta.
  - Samostalan, praktičan i kontinuirani rad!

# ORGANIZACIJA NASTAVE I PRAKTIČAN RAD

- Predavanja - 2 časa nedeljno
- Računske vežbe - 2 časa nedeljno
  - zadaci za razumevanje koncepata i uvežbavanje jednostavnih algoritamskih i programerskih primera
  - diskusija, demonstracije i konsultacije
- Laboratorijske vežbe - opciono
  - rešavanje zadataka u vezi sa nastavnim temama upotrebom računara
  - diskusija, demonstracije i konsultacije
- Praktičan samostalan rad:
  - domaći zadaci u toku semestra
  - rade se i brane samostalno, usmeno i na računaru
  - ulaze u konačnu ocenu

# OCENA ZNANJA

- U toku semestra student sakuplja ESPB bodove na osnovu svojih aktivnosti.
- Pri tome, njegova je obaveza, da bi mogao da polaže ispit, DA SAKUPI 30 POENA I ISPUNI SVOJE PREDISPITNE OBAVEZE.
- U suprotnom slučaju, iako je sakupio dovoljan broj poena za neku ocenu, predmet prenosi u drugu godinu i može ga polagati kada te obaveze ispuni.

| Zadatak | Broj poena |
|---|---|
| 55% dolazaka na vežbe i predavanja (8 termina od 15) | 30 |
| Kolokvijum 1 | 20 |
| Kolokvijum 2 | 20 |
| Završni ispit | 30 |

# OCENE

| Broj poena | Ocena |
|---|---|
| <=50 | 5 |
| 51-60 | 6 |
| 61-70 | 7 |
| 71-80 | 8 |
| 81-90 | 9 |
| 91-100 | 10 |

# LITERATURA


1. **PREZENTACIJE** sa web sajta Visoke tehničke škole.
2. **BELEŠKE** sa računskih vežbi.
3. dr Milan Popović, **OSNOVE PROGRAMIRANJA**
4. M. Tomašević, **ALGORITMI I STRUKTURE PODATAKA**, Akademska misao, Beograd, 2000.
5. S. Obradović, **VEŠTINA DOBROG PROGRAMIRANJA**, Visoka škola elektrotehnike i računarstva strukovnih studija, Beograd, 2004.
6. Prof. dr Dejan Živković, **UVOD U ALGIORITME I STRUKTURE PODATAKA,** Univerzitet Singidunum, Beograd, 2010.


# KOMUNIKACIJA

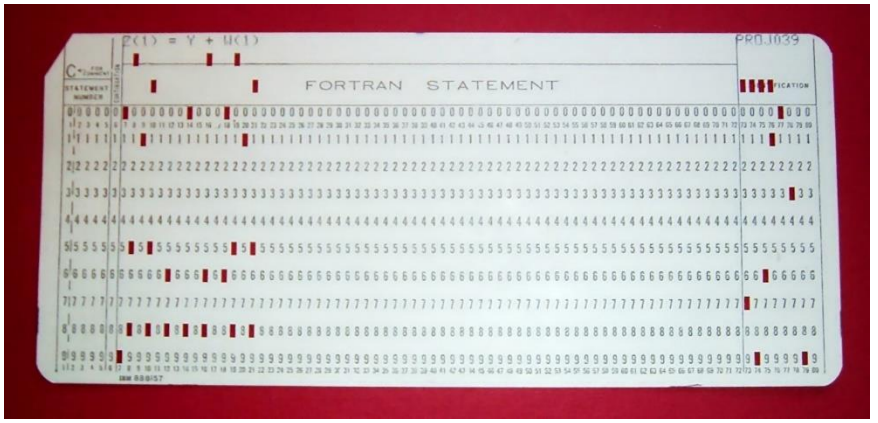

Microsoft Imagine

# PROGRAMIRANJE

# PROGRAMIRANJE

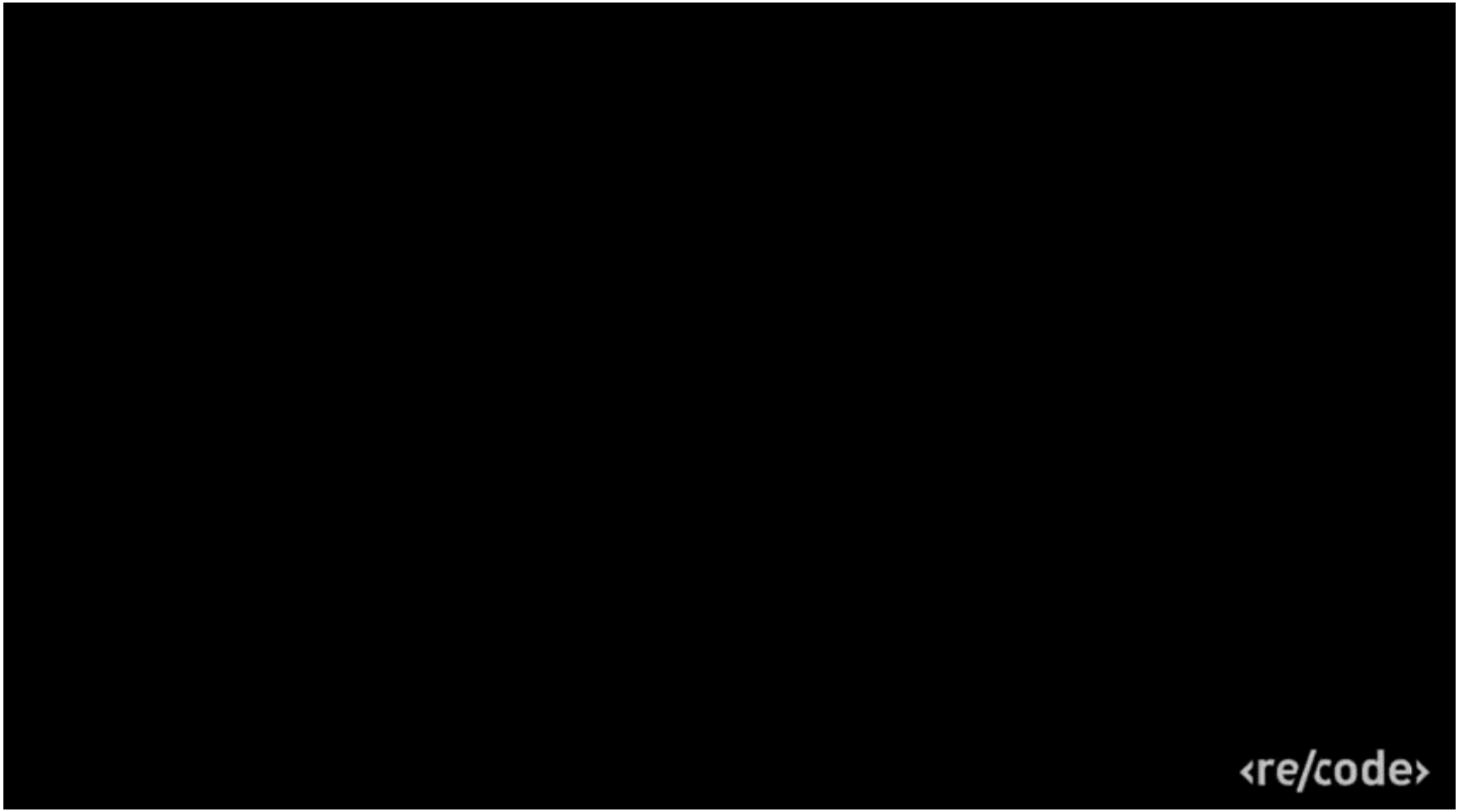
- PROGRAMIRANJE je proces zadavanja skupa naredbi u nekom programskom jeziku kako bi se izvršila neka aktivnost, odnosno, rešio određeni problem upotrebom računara.
- Da bi se neki proces mogao izvršavati na računaru potrebno je da on bude zapisan na nekom programskom jeziku koji odredeni računar „razume“.
- Programski jezik je notacija za precizan opis računarskih programa ili algoritama.
- Programski jezici su veštacki jezici, u kojima su sintaksa i semantika strogo definisani.
- Dakle, svrsishodni su ali ne dozvoljavaju slobodu izražavanja koja je svojstvena prirodnom jeziku.

# PROGRAMIRANJE NEKADA I SADA

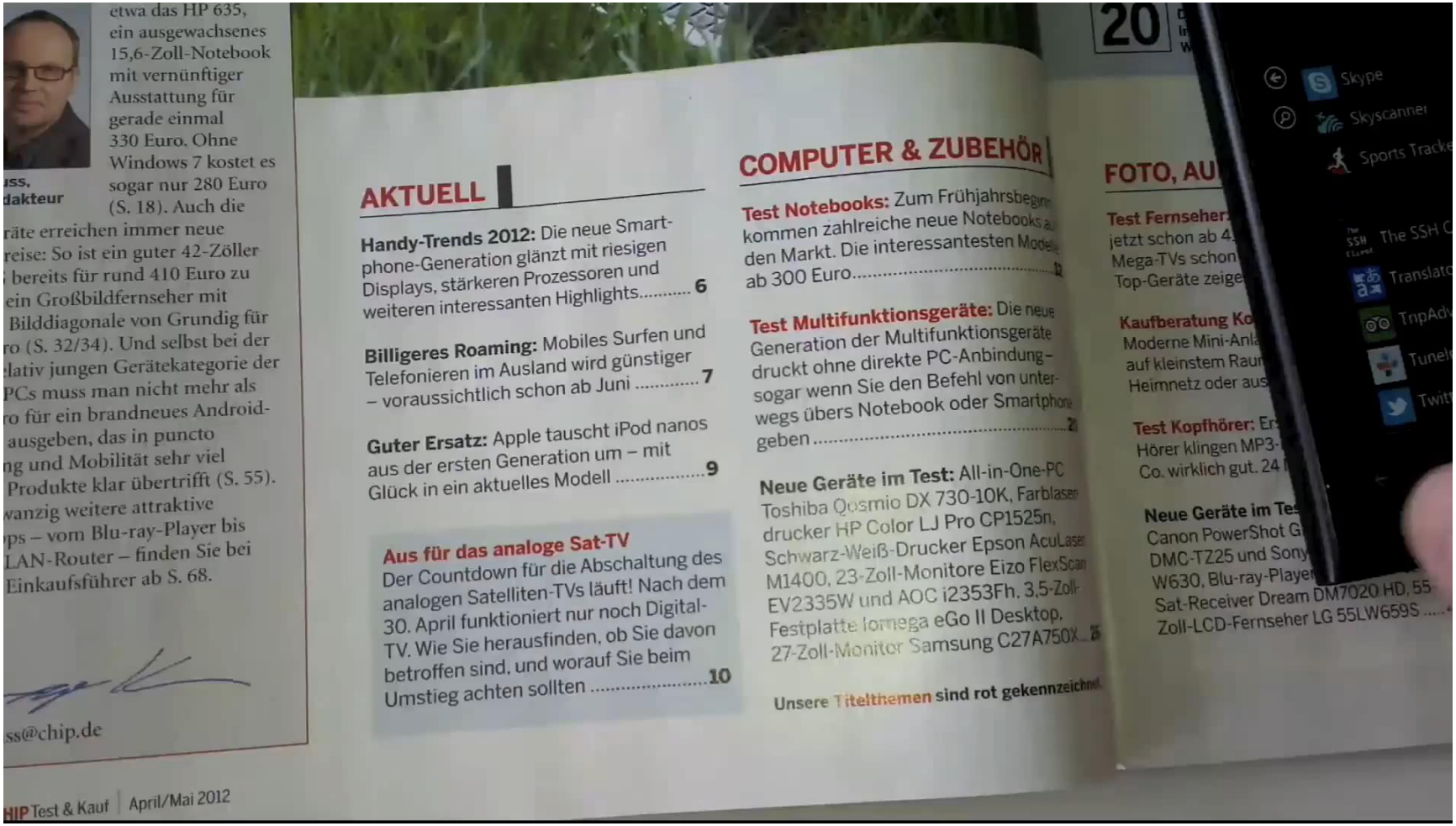



Bug?

# PRIMER I



# PRIMER 2



etwa das HP 635, ein ausgewachsenes 15,6-Zoll-Notebook mit vernünftiger Ausstattung für gerade einmal 330 Euro. Ohne Windows 7 kostet es sogar nur 280 Euro (S. 18). Auch die räte erreichen immer neue reise: So ist ein guter 42-Zöller bereits für rund 410 Euro zu ein Großbildfernseher mit Bilddiagonale von Grundig für ro (S. 32/34). Und selbst bei der lativ jungen Gerätekategorie der PCs muss man nicht mehr als ro für ein brandneues Android- ausgeben, das in puncto ng und Mobilität sehr viel Produkte klar übertrifft (S. 55). wanzig weitere attraktive ps – vom Blu-ray-Player bis LAN-Router – finden Sie bei Einkaufsführer ab S. 68.

ss@chip.de

HIP Test & Kauf | April/Mai 2012

## AKTUELL

**Handy-Trends 2012:** Die neue Smartphone-Generation glänzt mit riesigen Displays, stärkeren Prozessoren und weiteren interessanten Highlights..... 6

**Billigeres Roaming:** Mobiles Surfen und Telefonieren im Ausland wird günstiger – voraussichtlich schon ab Juni ..... 7

**Guter Ersatz:** Apple tauscht iPod nanos aus der ersten Generation um – mit Glück in ein aktuelles Modell ..... 9

**Aus für das analoge Sat-TV**
Der Countdown für die Abschaltung des analogen Satelliten-TVs läuft! Nach dem 30. April funktioniert nur noch Digital-TV. Wie Sie herausfinden, ob Sie davon betroffen sind, und worauf Sie beim Umstieg achten sollten ..... 10

## COMPUTER & ZUBEHÖR

**Test Notebooks:** Zum Frühjahrsbeginn kommen zahlreiche neue Notebooks auf den Markt. Die interessantesten Modelle ab 300 Euro.....

**Test Multifunktionsgeräte:** Die neue Generation der Multifunktionsgeräte druckt ohne direkte PC-Anbindung – sogar wenn Sie den Befehl von unterwegs übers Notebook oder Smartphone geben .....

**Neue Geräte im Test:** All-in-One-PC Toshiba Qosmio DX 730-10K, Farblaserdrucker HP Color LJ Pro CP1525n, Schwarz-Weiß-Drucker Epson AcuLaser M1400, 23-Zoll-Monitore Eizo FlexScan EV2335W und AOC i2353Fh, 3,5-Zoll-Festplatte Iomega eGo II Desktop, 27-Zoll-Monitor Samsung C27A750X...

Unsere Titelthemen sind rot gekennzeichnet.

## FOTO, AU

**Test Fernseher:** jetzt schon ab 4 Mega-TVs schon Top-Geräte zeige

**Kaufberatung Ko** Moderne Mini-Anla auf kleinstem Raur Heimnetz oder aus

**Test Kopfhörer:** Er Hörer klingen MP3- Co. wirklich gut. 24

**Neue Geräte im Te** Canon PowerShot G DMC-TZ25 und Sony W630, Blu-ray-Player Sat-Receiver Dream DM7020 HD, 55 Zoll-LCD-Fernseher LG 55LW659S .....

# PRIMER 3


IMPORTANT NOTICE
Nordeus uses cookies to give you the best experience. By using this site you agree to our use of cookies as described in this Cookie Policy.
11764582 USERS PLAYING ONLINE IN 42 LANGUAGES
Log In with Facebook
English
TOP ELEVEN
PLAY NOW
NEWS
FORUM
FAN KIT
SUPPORT
@sk
Help
Top Eleven Help
@TopElevenHelp
TOP Fast
TOP Accurate
TOP Informative
Help yourself faster and follow topeleven support on twitter now!
LATEST NEWS
TAKING OVER A NEW CLUB IN TOP ELEVEN
29 Sep 2014
Ever wanted to make a fresh start in Top Eleven? We have the perfect solution for you! This week, we will talk about the feature that allows you to take over a new club and to start a new career in the game – Top Eleven Job Center. All the Clubs seeking managers available to you will be listed in the Manager screen. The teams that you ... Read more ►
CHROME WEB STORE
Top Eleven is available on the Chrome Web Store
TOP ELEVEN LIVE
15. Oct - Matchday 23
16. Oct - Matchday 24
17. Oct - Matchday 25
15. Oct - 1/2 Finals - 2nd Leg

# GDE SVE RADE PROGRAMERI?

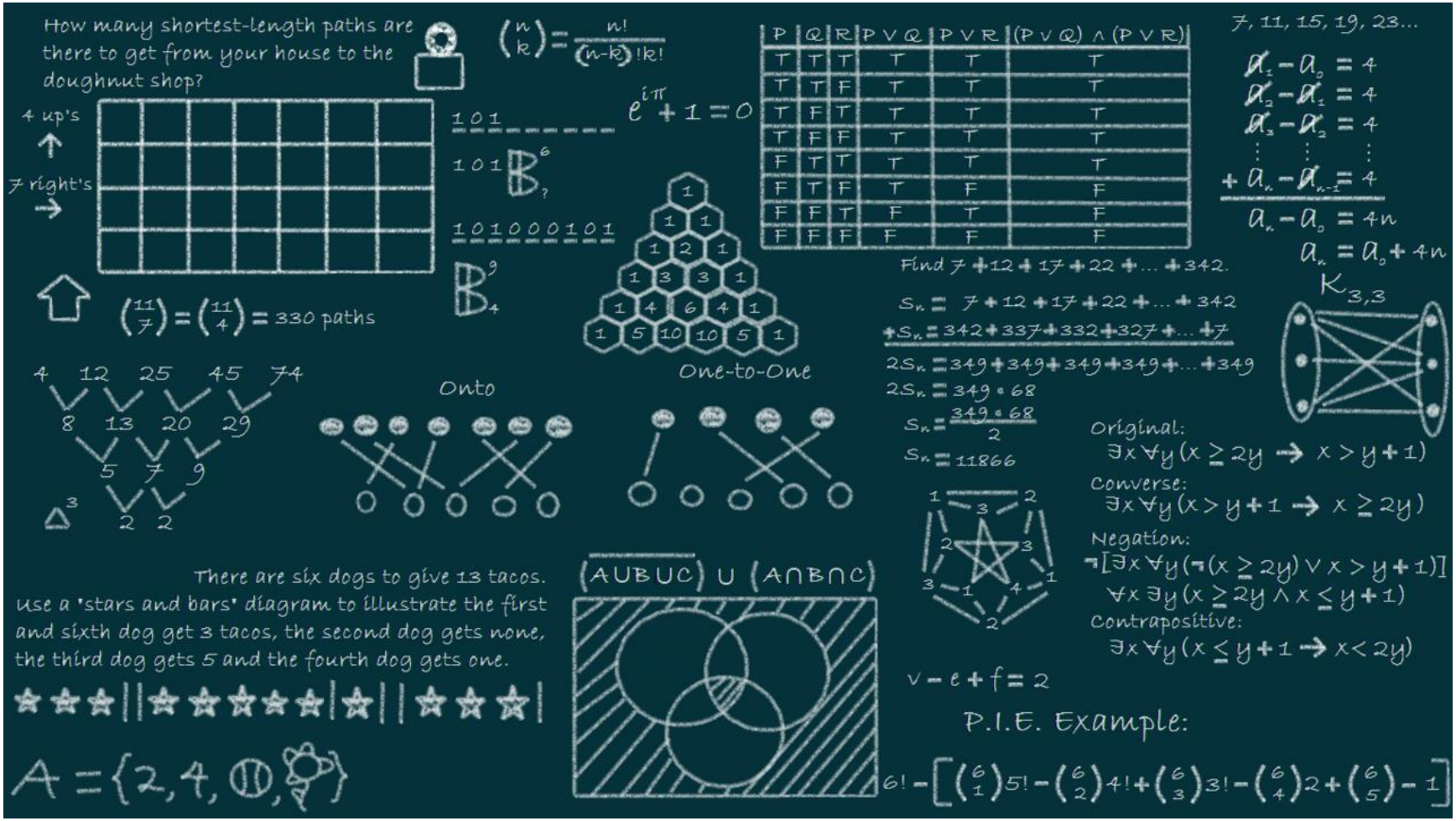
- Backend Developer
- Frontend Developer
- Help Desk Technician
- Project Manager
- Mobile Developer
- Network Engineer
- Network Architect
- Security Engineer
- Database Administrator
- Server Administrator
- System Analyst
- Business Analyst
- System Architect
- Web Designer
- Quality Assurance
- Game Developer


Construction 100 $
Marketing 97 $
HR 89 $
Logistic 60 $
Consulting 555
IT costs
Share of the income of participants by quarter
Quarter I
Quarter II
Quarter III
Quarter IV
Daily income per employee per hour by department
Logistics costs
IT 105 $
Construction 100 $
Marketing 97 $
HR 89 $
Logistic 60 $
Consulting 555
FRT division
GHT 254 550 254 274 154 415
RDW 650 320 754 273 825 154

# ALGORITMI I STRUKTURE PODATAKA

- PREDAVANJE 2 -

# ALGORITMI


How many shortest-length paths are there to get from your house to the doughnut shop?
4 up's
7 right's
$\binom{11}{7} = \binom{11}{4} = 330$ paths
$\binom{n}{k} = \frac{n!}{(n-k)!k!}$
$e^{i\pi} + 1 = 0$
101
1010000101
1 1 1 1 2 1 1 3 3 1 1 4 6 4 1 1 5 10 10 5 1
4 12 25 45 74
8 13 20 29
5 7 9
2 2
Onto
One-to-One
P Q R P∨Q P∨R (P∨Q)∧(P∨R)
T T T T T T
T T F T T T
T F T T T T
T F F T T T
F T T T T T
F T F T F F
F F T F T F
F F F F F F
Find 7 + 12 + 17 + 22 + ... + 342.
$S_n = 7 + 12 + 17 + 22 + \dots + 342$
$+S_n = 342 + 337 + 332 + 327 + \dots + 7$
$2S_n = 349 + 349 + 349 + 349 + \dots + 349$
$2S_n = 349 \cdot 68$
$S_n = \frac{349 \cdot 68}{2}$
$S_n = 11866$
7, 11, 15, 19, 23...
$a_1 - a_0 = 4$
$a_2 - a_1 = 4$
$a_3 - a_2 = 4$
$a_n - a_{n-1} = 4$
$a_n - a_0 = 4n$
$a_n = a_0 + 4n$
$K_{3,3}$
Original:
$\exists x \forall y (x \geq 2y \rightarrow x > y+1)$
Converse:
$\exists x \forall y (x > y+1 \rightarrow x \geq 2y)$
Negation:
$\neg[\exists x \forall y (\neg(x \geq 2y) \vee x > y+1)]$
$\forall x \exists y (x \geq 2y \wedge x \leq y+1)$
Contrapositive:
$\exists x \forall y (x \leq y+1 \rightarrow x < 2y)$
There are six dogs to give 13 tacos. Use a 'stars and bars' diagram to illustrate the first and sixth dog get 3 tacos, the second dog gets none, the third dog gets 5 and the fourth dog gets one.
$(A \cup B \cup C) \cup (A \cap B \cap C)$
$v - e + f = 2$
P.I.E. Example:
$6! - \left[\binom{6}{1}5! - \binom{6}{2}4! + \binom{6}{3}3! - \binom{6}{4}2 + \binom{6}{5} - 1\right]$

# POJAM ALGORITMA

- **Algoritam** je konačan niz jasno definisanih koraka koji za date početne uslove vode do rešenja.
- **Algoritam** je skup nedvosmislenih pravila u cilju rešavanja određenog tipa zadatka.

- Svako pojedinačno pravilo zove se **algoritamski korak**.
- Algoritmi su veoma bliski računarskim programima i ponekad se sa njima poistovećuju.
- Međutim, algoritmom se mogu opisati mnogi postupci koji se nikada neće izvršavati putem računara.

# POJAM ALGORITMA

- Algoritam je konačan niz naredbi, od kojih svaka ima jasno značenje i može se izvršiti u konačnom vremenu.

- Izvršavanjem tih naredbi
  **zadani ulazni podaci** pretvaraju se u **izlazne podatke (rezultate)**.

- **Pojedine naredbe mogu se izvršavati uslovno**, ali u tom slučaju same naredbe moraju opisati uslove izvršavanja.

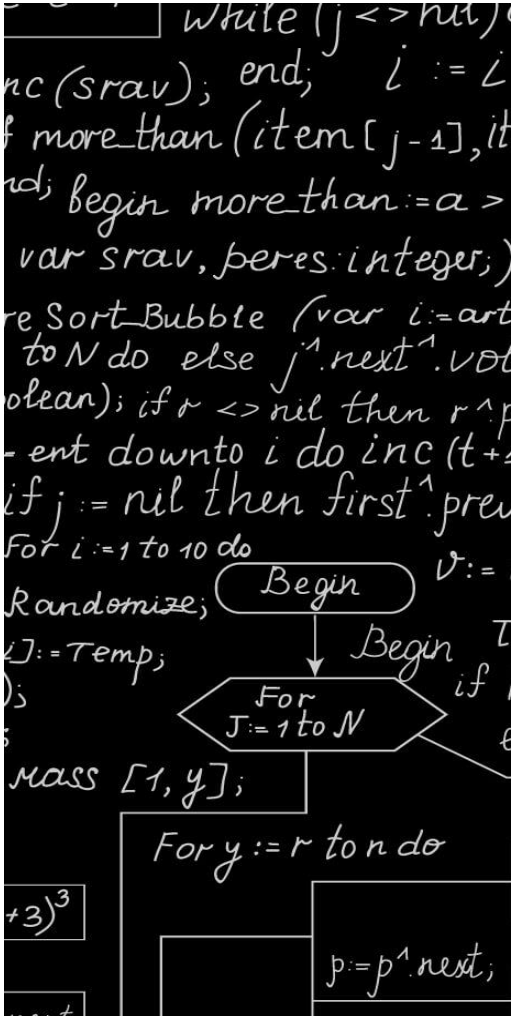
- Takođe, **iste naredbe mogu se izvršiti više puta**, pod pretpostavkom da same naredbe ukazuju na ponavljanje.

- Ipak, zahtevamo da za bilo koje vrednosti ulaznih podataka algoritam završava nakon konačnog broja ponavljanja.

# VAŽNE OSOBINE ALGORITMA

- Važne osobine:
  - jasan i precizan
  - konačan broj koraka

# PRIKAZ ALGORITMA




- Postoji više načina zapisivanja algoritma:
  - korišćenjem govornog jezika
  - pseudokod
  - dijagrami toka
  - programski jezik

# PRIKAZ ALGORITMA DEFINISANOG GOVORNIM JEZIKOM

- Da bi algoritam koji saopštavamo prirodnim jezikom bio precizan i dovoljno detaljan potrebno je voditi računa da izlaganje bude jasno i nedvosmisleno, što je posebno značajno pri izlaganju redosleda operacija koje se moraju izvršiti.

- ZADATAK: Napisati govornim jezikom algoritam za spremanje voćnog kolača.

# PRIKAZ ALGORITMA DEFINISANOG GOVORNIM JEZIKOM

- Izaberi voće
- Očisti voće od nejestivih delova
- Iseckaj voće na što sitnije komade
- Poređaj voće u posudu
- Umuti jaja
- Dodaj vanilin šećer i umuti
- Dodaj kristal šećer i umuti
- Dodaj brašno i umuti
- Dodaj mleko i umuti
- Ponavljaj
  - peci u rerni na 180-200 stepeni

Sve dok ne požuti i ne stegne se i ne zamiriše.

# PRIMER ALGORITMA DEFINISANOG PSEUDO JEZIKOM

- Pseudo jezik je neformalna kombinacija prirodnog jezika i nekog zamišljenog programskog jezika, pa njegova upotreba podrazumeva zapisivanje algoritama u obliku koji je nalik na neki programski jezik.

- ZADATAK: Napisati algoritam pseudokodom za pronalaženje najvećeg od tri zadana realna broja: x, y, z.
- Primer pseudokoda koji koristi isključivo termine govornog jezika:
  - pročitaj tri realna broja
  - ispiši pročitane brojeve
  - odredi najveći broj
  - ispiši nađeni broj

# PRIMER ALGORITMA DEFINISANOG PSEUDO KODOM

- Pseudokod koji koristi uobičajene simbole

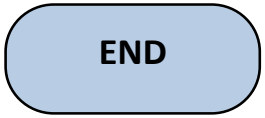
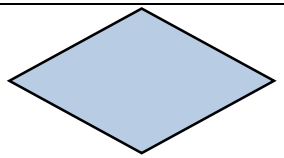
```
pročitaj (x,y,z)
ispiši (x,y,z)
/odredi najveći broj/
rez = z  /uvođenje pomoćne promenljive/
(ako je x > y tada
          ako je x > z tada rez = x )
inače
(ako je y > z tada rez = y )

ispiši (rez)
kraj
```

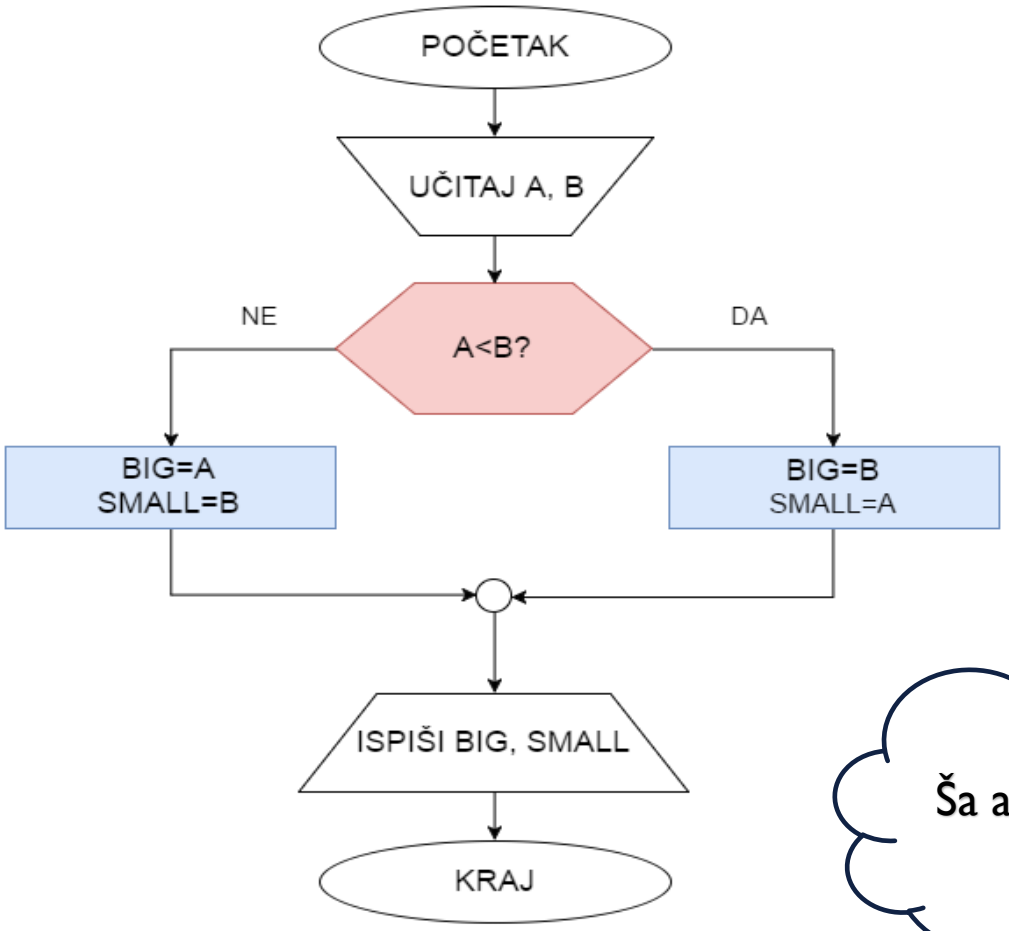
# PRIKAZ ALGORITMA POMOĆU DIJAGRAMA TOKA

- Da bi zapis algoritma bio pregledan, pogodno je koristiti njegov grafički prikaz. To se zove **algoritamska šema** (skraćeno algoritam), blok dijagram ili dijagram toka.
- Algoritmi se za potrebe izvršavanja na računaru zapisuju u obliku programa na nekom programskom jeziku.
- Da bi se neki algoritam mogao izvršavati na računaru potrebno je da on bude zapisan na nekom programskom jeziku koji određeni računar „razume“.
- Programski jezici su veštački jezici, u kojima su sintaksa i semantika strogo definisani. Dakle, svrsishodni su ali ne dozvoljavaju slobodu izražavanja koja je svojstvena prirodnom jeziku.

# GRAFIČKI ELEMENTI

| | |
|---|---|
|   | Blok za početak algoritamske šeme (dijagrama) |
|   | Blok za završetak algoritamske šeme (dijagrama) |
|  | Blok za ulazne podatke algoritma |
|  | Blok za izlazne podatke – rezulatate algoritma |
|  | Blok za obradu podataka |
|  | Uslovni blok – sadrži uslov na osnovu koga se određuje dalji tok algoritma |

# PRIKAZ ALGORITMA POMOĆU DIJAGRAMA TOKA




Ša ako je A = B?

# PROMENLJIVE VELIČINE U ALGORITMU

- Promenljiva je prostor u memoriji računara koji je imenovan i služi za čuvanje odgovarajućih vrednosti.

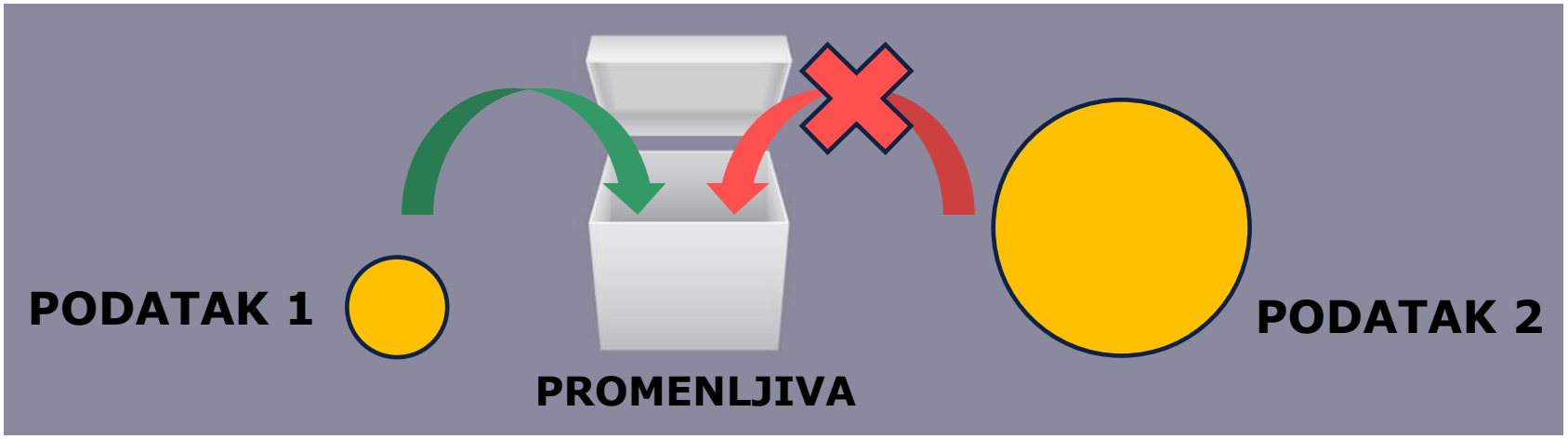
- Pri izradi algoritama promenljivim se daju imena (oznake) po želji, bitno je samo da se ta imena mogu zapisati u okviru zapisa algoritma.

- U programskim jezicima imena se obično ograničavaju na slova latinice i još neke dopunske simbole.

# STRUKTURA PODATAKA

- Struktura podataka predstavlja skup promenljivih u nekom programu zajedno sa vezama izmedu njih.

- Stvorena je sa namerom da omogući čuvanje određenih podataka i efikasno izvršavanje određenih operacija nad tim podacima (upisivanje, promena, čitanje, traženje po nekom kriterijumu, brisanje. . . ).



# OPERATORI

- Operatori su simboli koji označavaju određenu operaciju i koji povezuju jedan ili više operanada u izraz.

Podela po:

- broju operanada
- po funkcionalnosti

U zavisnosti od broja operanada operatori u se dele na:

- unarne
- binarne
- ternarne

# OPERATORI

- **Grupe operatora u odnosu na funkcionalnost:**
  - operatori dodeljivanja vrednosti
  - aritmetički operatori
  - relacioni operatori
  - logički operatori

# OPERATOR DODELE

- Ovo je binarni operator koji smešta vrednost desnog operanda na lokaciju čije se ime nalazi sa leve strane operatora.
- Levi operand mora biti ime memorijske lokacije.

**ime memorijske lokacije = vrednost**

X = 5

X = 5 + 4

X = Y – 23

X = X + 5

# ARITMETIČKI OPERATORI

- Imaju numeričke operande i rezultati su, takođe, numeričkog tipa.

- Po prioritetu:
  - Unarni operatori + i –
  - Operatori inkrementiranja (++) i dekrementiranja (--)
  - Operatori *, /, %
  - Binarni operatori + i -

# RELACIONI OPERATORI

- Ovo su operatori za poređenje vrednosti operanada.
- Ovoj grupi operatora pripadaju:

< (manje),

<= (manje ili jednako),

> (veće),

>= (veće ili jednako),

== (jednako),

!= (različito).

5 < 7 rezultat je?
5 > 7 rezultat je?
7 <= 7 rezultat je?
8 >= 7 rezultat je?
5 == 5 rezultat je?
5 == 7 rezultat je?
5.0 == 5 rezultat je?
6 != 5 rezultat je?

- Prioritet prva 4 operatora je viši od prioriteta poslednja dva.

# LOGIČKI OPERATORI

- Obzirom da se ovi operatori prirodno (u matematici) primenjuju nad logičkim podacima, pri izvršenju ovih operacija svaka vrednost različita od nule se tretira kao logička vrednost tačno (true), jedino 0 označava netačno (false).
- U ovu grupu operatora spadaju:

  `!` (negacija),

  `&&` (konjukcija, tj. logičko I),

  `||` (disjunkcija, tj. logičko ILI).
- Rezultat operacije je 1 ako je rezultat logičke operacije tačno,
- u suprotnom, rezultat je 0.

# OSOBINE ALGORITAMA

- **1. Diskretnost algoritama -** Ako posmatramo vreme izvršavanja algoritma, svakom koraku možemo pridružiti diskretan vremenski period u kome se taj korak izvršava.

- **2. Determinisanost -** Svaki korak sadrži ulazne veličine, na osnovu kojih se jednoznačno dobijaju izlazne veličine.

- **3. Elementarnost -** Zakon dobijanja izlaznih veličina mora biti jasan i jednostavan.

- **4. Rezultativnost -** Za svaki skup ulaznih veličina mora biti definisano šta je rezultat.

- **5. Masovnost -** Algoritam treba tako napraviti da važi za najširi skup ulaznih podataka (rešavanje opštih problema ).

# FORMIRANJE ALGORITMA

- Postoji više metoda formiranja algoritma, ali mi ćemo se zadržati na metodu od vrha naniže (top-down).
- Metod formiranja algoritama od vrha prema dnu podrazumeva postepeno rešavanje detalja u polaznom vrlo uopštenom algoritmu.
- Rešavanje po nivoima omogućava izgradnju složenog algoritma iz više jednostavnijih, čime se postupak čini jednostavnijim i bržim i smanjuje se verovatnoća pojavljivanja grešaka.
- Konstrukcija algoritma se završava kada svi procesi budu do kraja razvijeni, odnosno opisani dovoljno detaljno da se algoritam može implementirati.
- Ovaj metod se dobro uklapa u strukturno programiranje i kao svoj završni rezultat daje program na nekom programskom jeziku.

# FORMIRANJE ALGORITMA

- Pri formiranju odgovarajućeg algoritma imamo sledeće korake:

1. **Razumevanje problema –** Kada formiramo algoritam, odnosno rešavamo neki problem najpre moramo razumeti problem.

   Prvo nalazimo šta se od nas traži, šta nam je sve dostupno od informacija vezanih za naš problem i zatim razmišljamo o pravilima na osnovu kojih moramo rešiti problem.

2. **Zapisivanje na prirodnom jeziku ideja za rešavanje problema –** Kada uspemo da formiramo neko rešenje od trenutnih ideja koje imamo, od njega pravimo model, odnosno formalnije i preciznije zapisujemo korake ka rešenju problema.

3. **Izgradnja modela (pisanje matematičkih formula, izraza) –** podrazumeva pisanje formula i izraza koji nam opisuju problem i pomažu u dobijanju željenih izlaznih rezultata

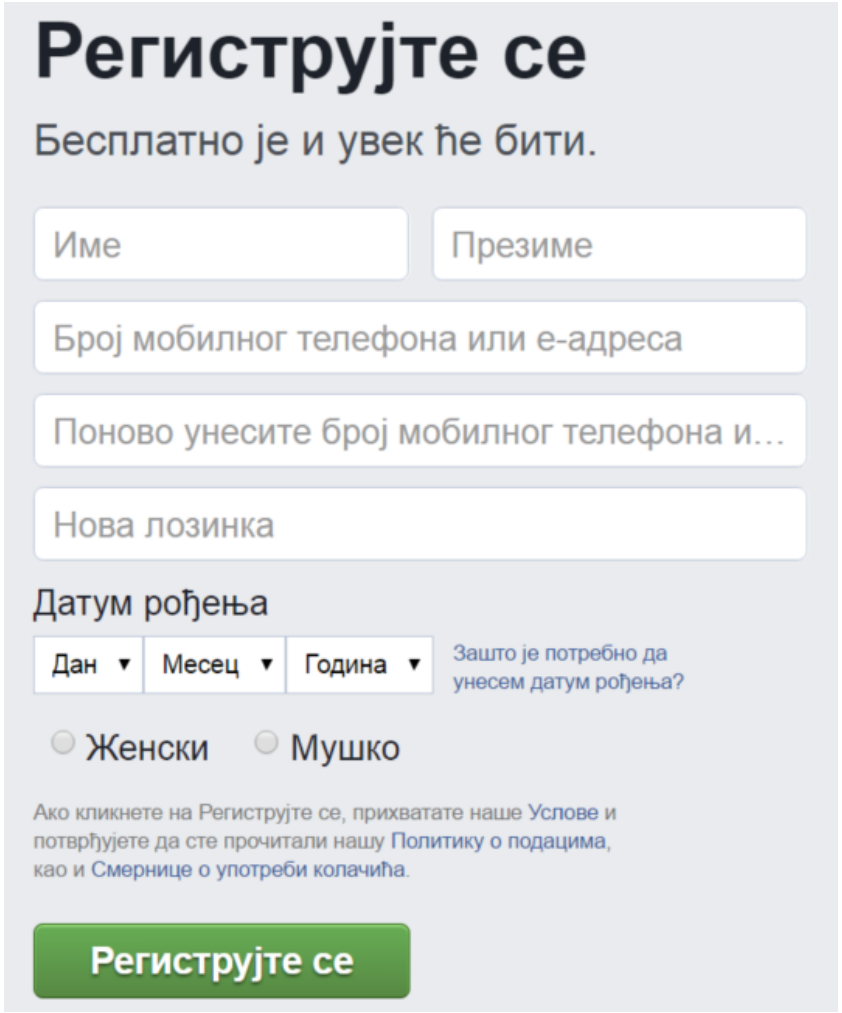
4. **Formulisanje algoritma –** Na osnovu modela formiramo algoritam.

# ZADATAK – FORMIRATI FACEBOOK PROFIL FAZA I – RAZUMEVANJA PROBLEMA

- Šta znači napraviti Fejsbuk profil?

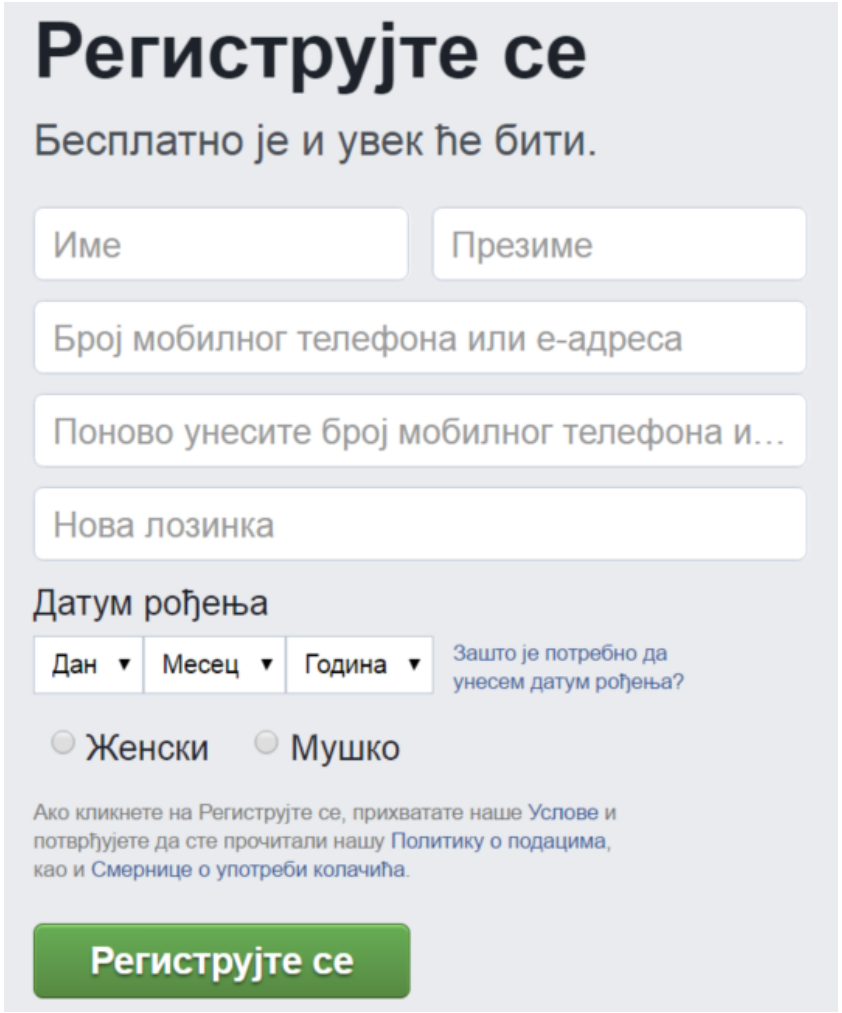
- Otići na www.facebook.com

- Popuniti polja u formularu za registraciju i kliknuti na **"Registruj se"**



# ZADATAK – FORMIRATI FACEBOOK PROFIL
# FAZA 2 – ZAPISIVANJE NA PRIRODNOM JEZIKU

- Odlazak na www.facebook.com
- Unos imena - Pera
- Unos prezimena - Petrović
- **Unos e-mail adrese**
  - Pitanje, da li imam e-mail adresu?
    - NEMAM – napravim je, unesem
    - IMAM - unesem je
- Unesem lozinku – nek@Lozink@
- Izaberem pol - Muško
- Izaberem datum rođenja - 4.4.1981
- Kliknem na "Registruj se"



# ZADATAK – FORMIRATI FACEBOOK PROFIL
# FAZA 3 – IZGRADNJA MODELA

- Imenovani prostor u memoriji računara znači da će promenljiva dobiti svoje mesto u memoriji, na kome će pisati njeno ime.
- Pri unosu podataka u program, podatke smeštamo u promenljive, kako bi mogli da ih koristimo u toku rada programa.
- Tako da, kada napišemo **Ime**=? , mi zapravo tražimo unos neke vrednosti za promenljivu Ime.
- Pisanjem **Ime = Pera** promenljivoj **Ime** dodeljujemo vrednost **Pera**.
- Unutar memorije računara se na mestu koje pripada promenljivoj **Ime** upisuje vrednost **Pera**.
- Promenljivima dodeljujemo vrednosti pri unosu i pri izvršavanju programa.

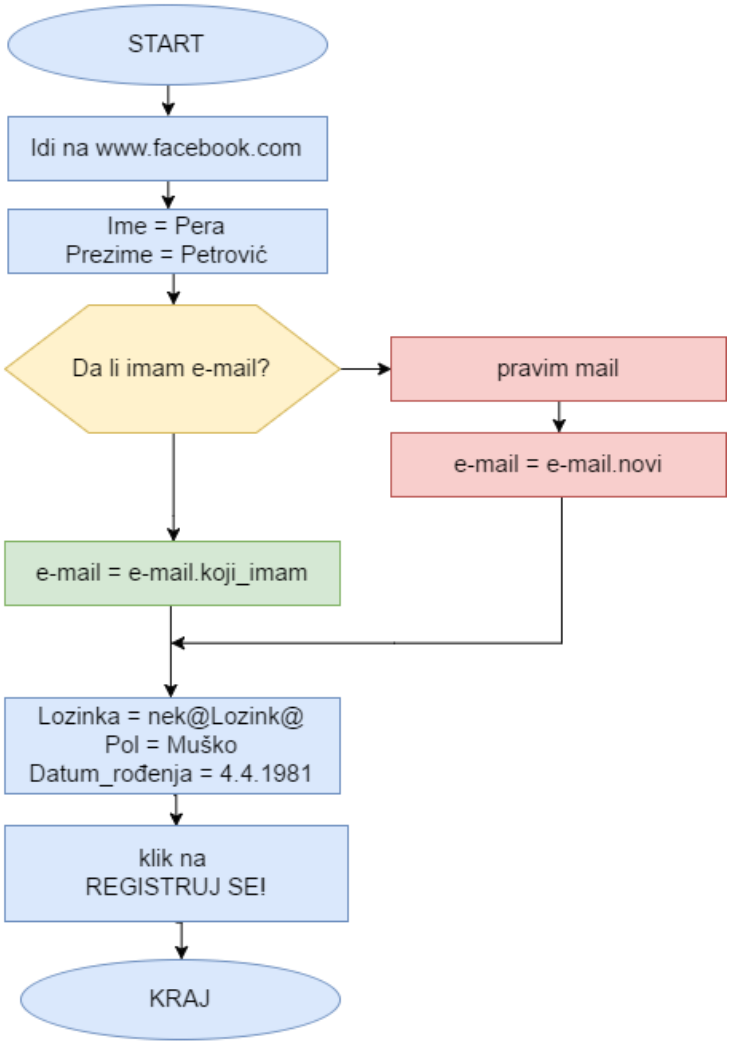
IME PROMENLJIVE = VREDNOST PROMENLJIVE

# ZADATAK – FORMIRATI FACEBOOK PROFIL FAZA 3 – IZGRADNJA MODELA

- Ime=?
- Prezime=?
- e-mail=?
- Lozinka=?
- Pol=?
- Datum_rođenja=?

- Ime = Pera
- Prezime = Petrović
- Da li imam e-mail? IMAM/NEMAM
  - NEMAM
    - Pravljenje e-mail adrese
    - e-mail = email.novi
  - IMAM
    - e-mail = email.koji_imam
- Lozinka = nek@Lozink@
- Pol = Muško
- Datum_rođenja = 4.4.1981
- Svi koraci završeni, klik na **Registruj se**!!!

# ZADATAK – FORMIRATI FACEBOOK PROFIL
# FAZA 4 – FORMULISANJE ALGORITMA



# ELEMENTARNE STRUKTURE ALGORITAMA

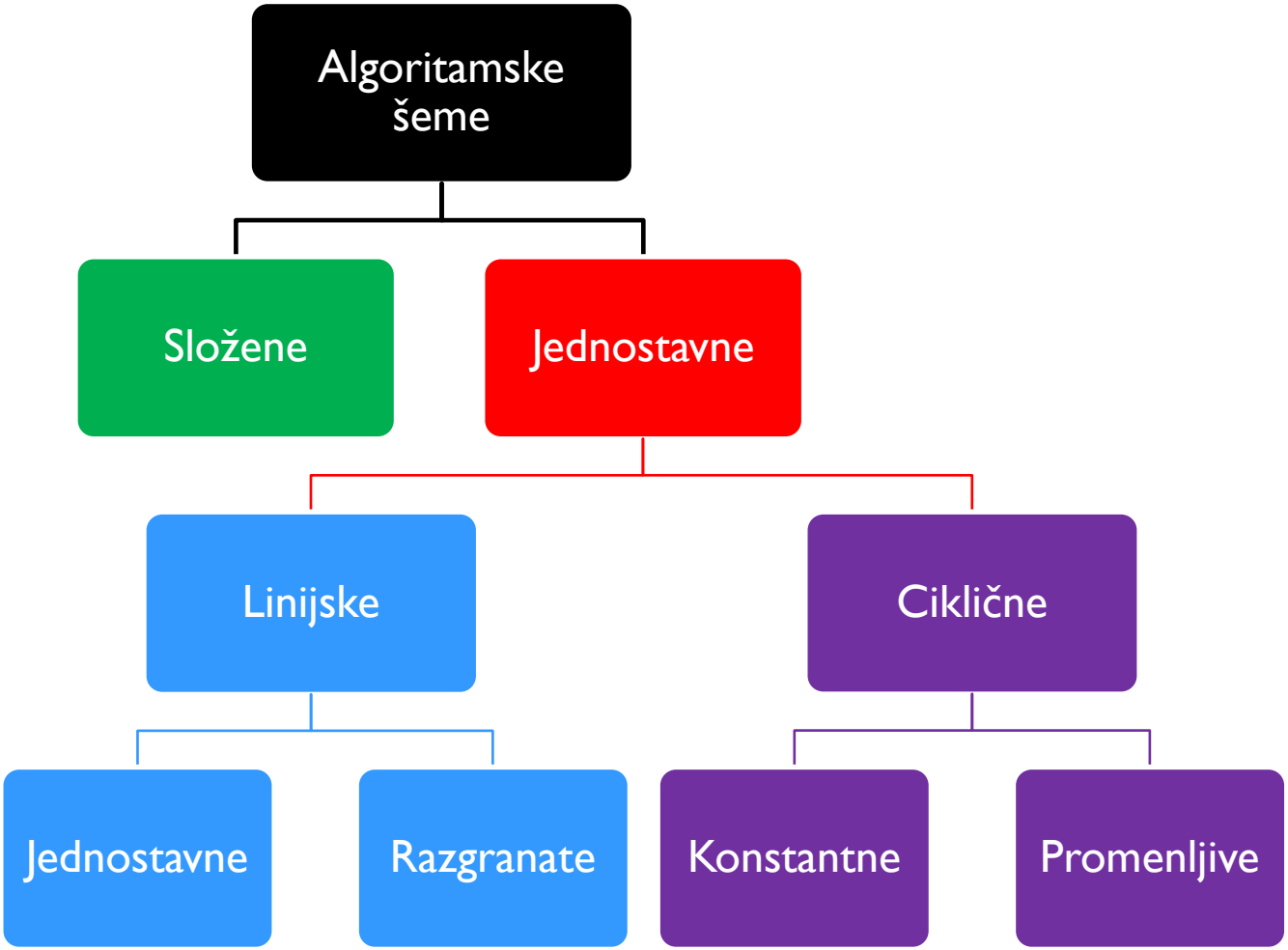
- Pod algoritamskom (programskom) strukturom podrazumevamo više koraka (komandi programskog jezika) koji čine jednu celinu.
- Postoje tri elementarne algoritamske strukture:
  - Linijska
  - Razgranata
  - Ciklična

# ELEMENTARNE ALGORITAMSKE STRUKTURE

- Značenje ovih elementarnih algoritamskih struktura je:
  - Linijska - sve akcije se izvršavaju tačno jednom u redosledu u kome su navedene
  - Razgranata (SELEKCIJA) - omogućuje da se od više grupa akcija, koje se nalaze u različitim granama razgranate strukture, izabere ona koja će se izvršiti jednom, dok se sve ostale grupe akcija neće izvršitini nijednom
  - Ciklična (ITERACIJA) - skup akcija može se izvršiti više puta

- Algoritamsko rešenje bilo kog problema može se uvek zapisati korišćenjem samo ove tri strukture.

# VRSTE ALGORITAMA



# ALGORITMI I STRUKTURE PODATAKA

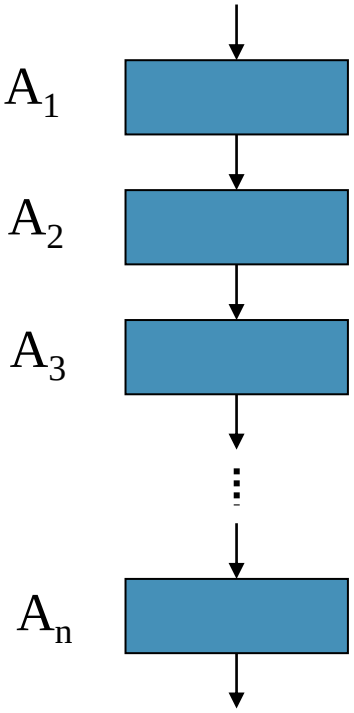
- PREDAVANJE 3 -

# 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

- 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*

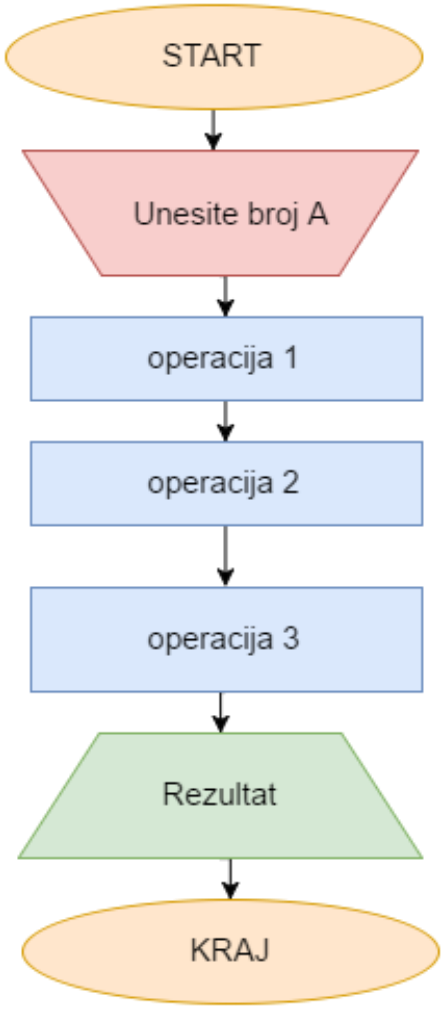
  *a = b*

  *n = n+1*

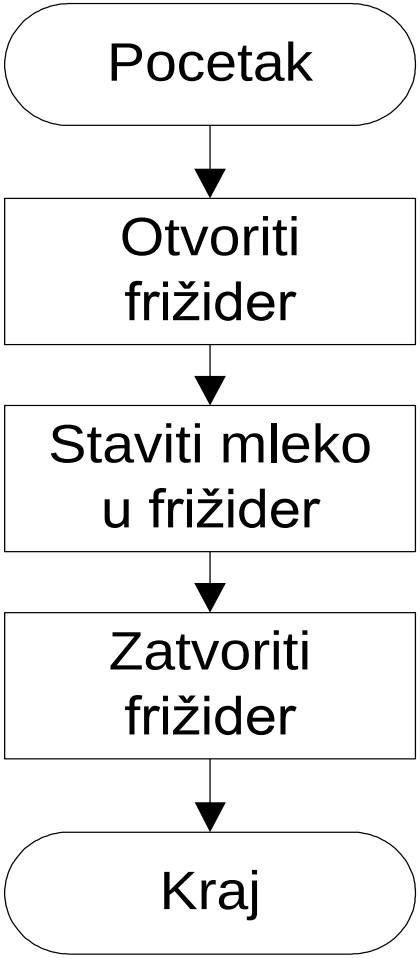
# LINIJSKA ALGORITAMSKA STRUKTURA

- Sekvenca je niz algoritamskih koraka koji se **bezuslovno** izvode jedan za drugim.

- Linijska algoritamska struktura ima tačno jednu ulaznu tačku, tačno jednu izlaznu tačku i takav tok da se svaki njen korak bezuslovno izvršava tačno jednom.



# LINIJSKA ALGORITAMSKA STRUKTURA




(1) Otvoriti frižider;

(2) Staviti mleko u frižider;

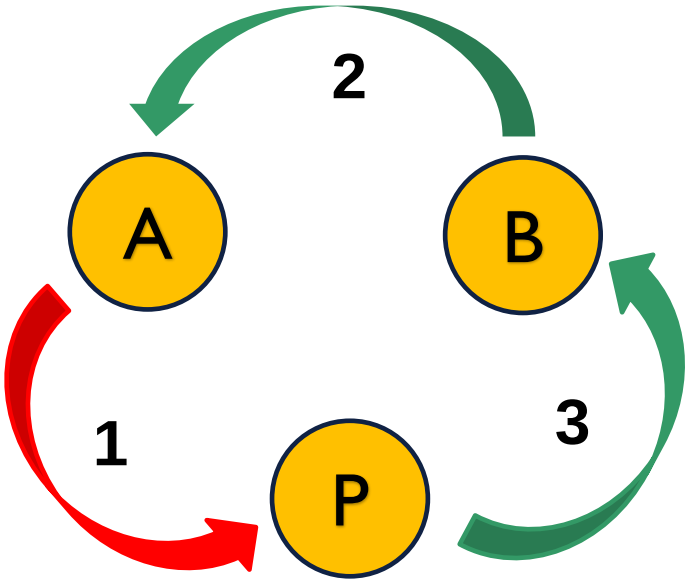
(3) Zatvoriti frižider.

# LINIJSKA ALGORITAMSKA STRUKTURA PRIMER 1

Zameniti sadržaje dve memorijske lokacije, A i B

- Rešenje – potrebna je treća, pomoćna, lokacija

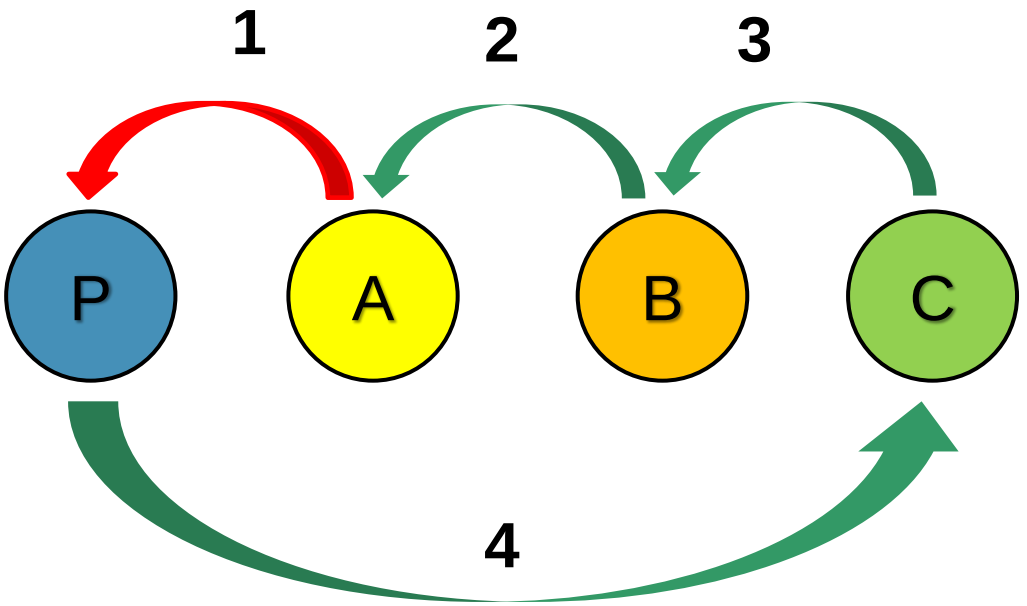
1. sadržaj lokacije A zapamtiti u pomoćnu lokaciju, P
2. sadržaj lokacije B zapamtiti u lokaciju A
3. sadržaj lokacije P zapamtiti u lokaciju B
4. kraj



# LINIJSKA ALGORITAMSKA STRUKTURA PRIMER 2

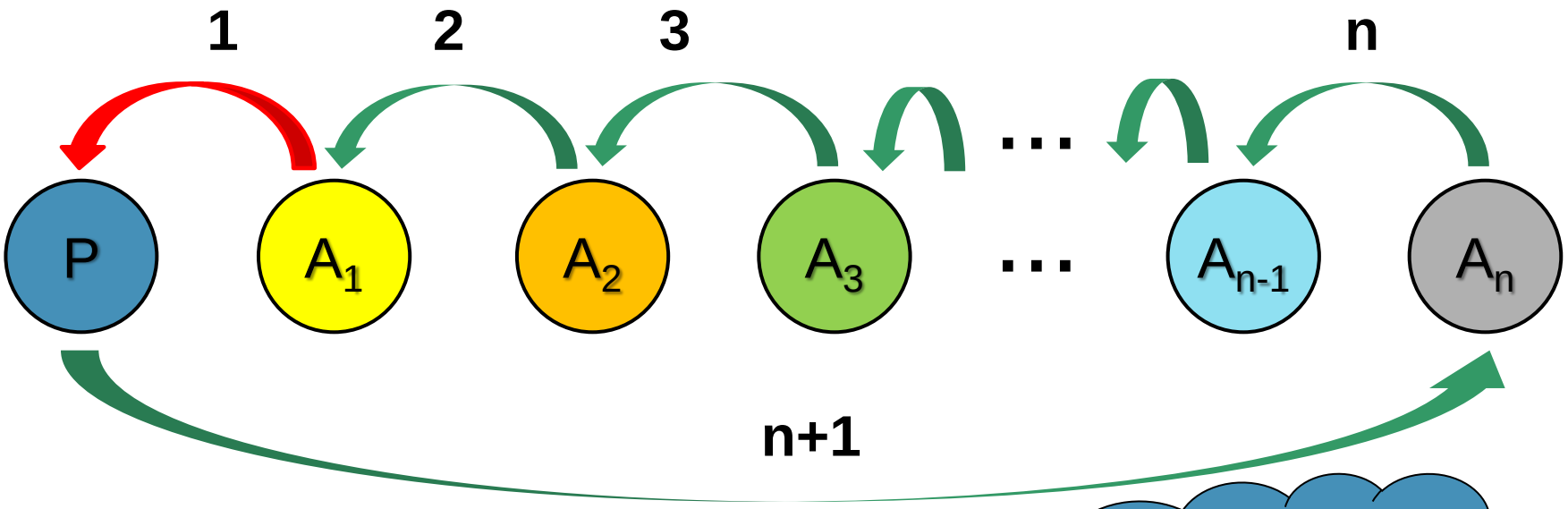
- Ciklično pomeriti ulevo sadržaje lokacija A, B i C

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



# LINIJSKA ALGORITAMSKA STRUKTURA PRIMER 3

- Ciklično pomeriti ulevo sadržaje lokacija $A_1, A_2, A_3, \ldots A_n$



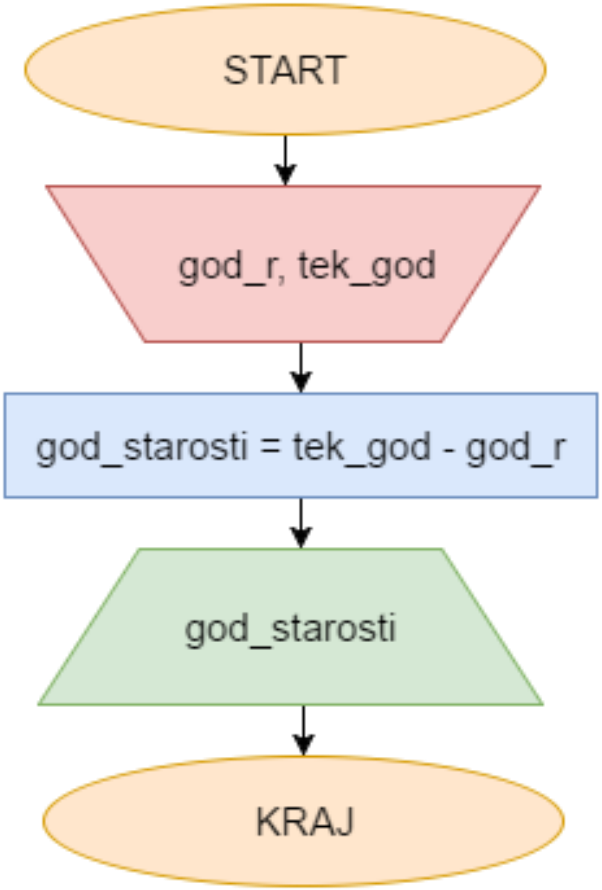

- iz $A_1$ u P
- iz $A_i$ u $A_{i-1}$, za i=2,...,n
- iz P u $A_n$
- kraj

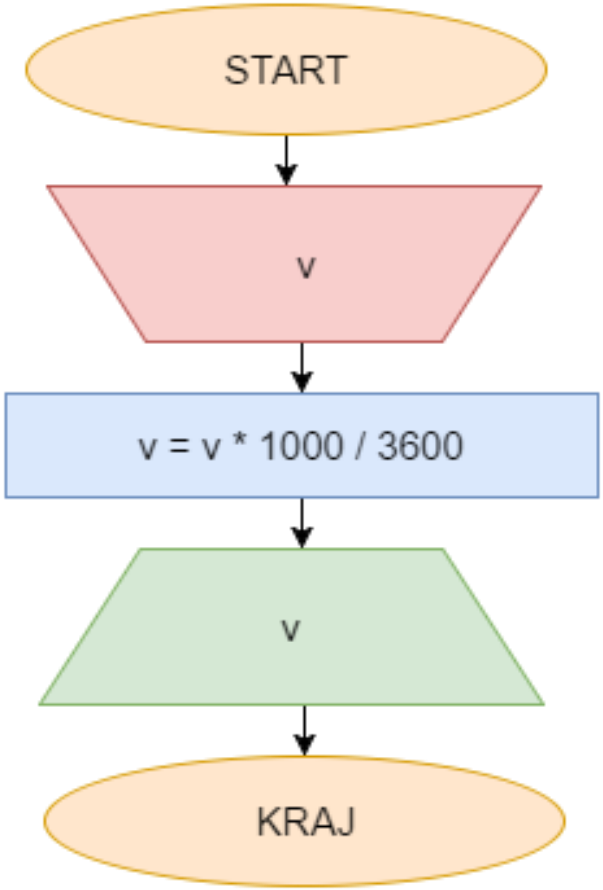
kako izvršiti pomeranje za k mesta?

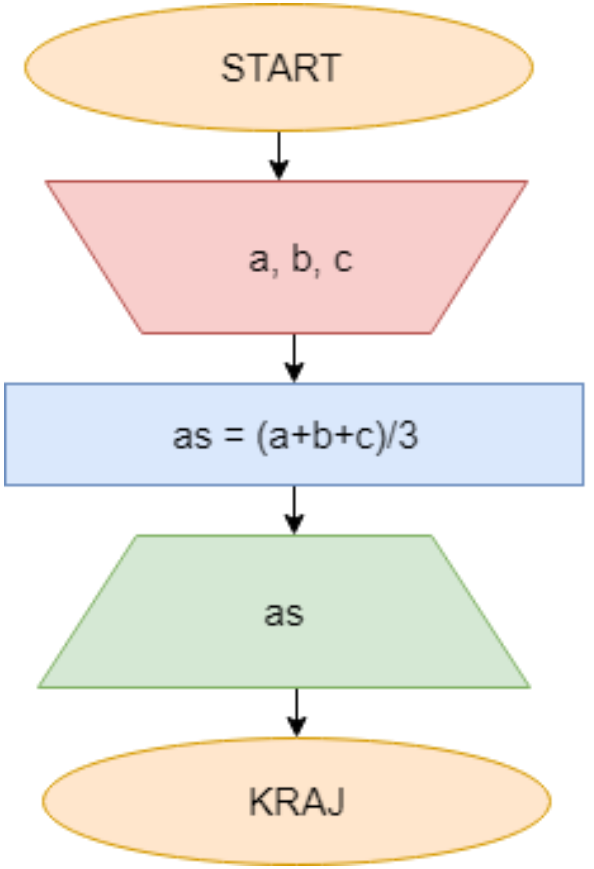
# LINIJSKA ALGORITAMSKA STRUKTURA PRIMERI

- Napisati algoritam za računanje godina starosti.
- Napisati algoritam za pretvaranje km/h u m/s.
- Naći aritmetičku sredinu tri broja.

# LINIJSKA ALGORITAMSKA STRUKTURA PRIMERI



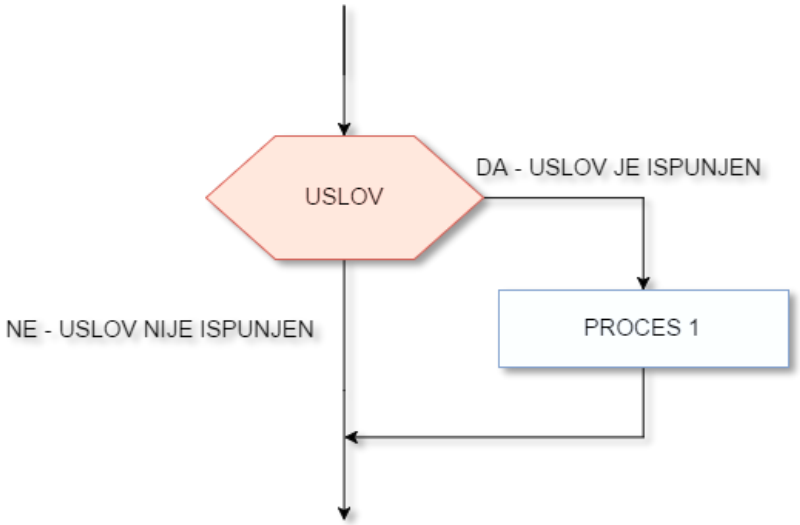



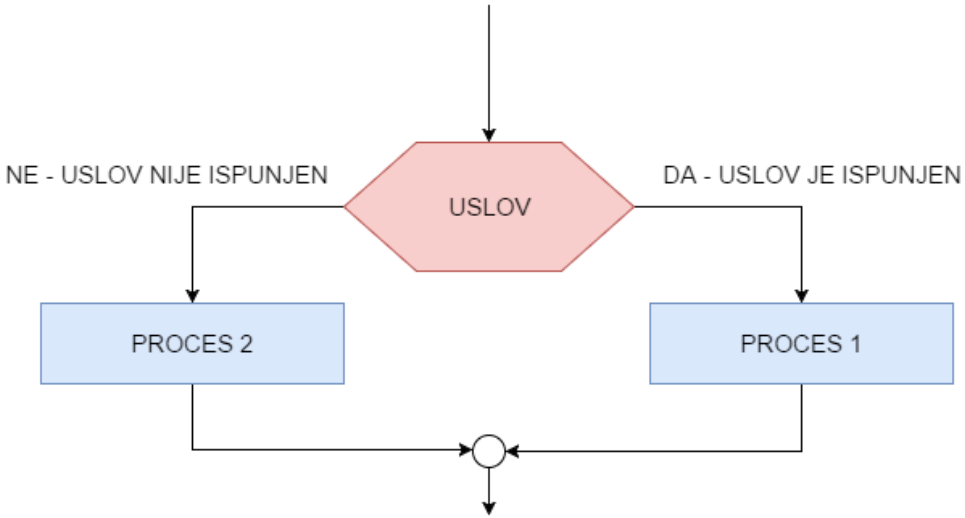
# LINIJSKA ALGORITAMSKA STRUKTURA PRIMERI

- DOMAĆI ZADATAK: Za poznate katete naći obim i površinu pravouglog trougla.

# ALTERNACIJA (SELEKCIJA)

- Omogućava uslovno izvršenje niza algoritamskih koraka
- Blokovi označeni sa PROCES1 i PROCES2 mogu sadržati bilo koju kombinaciju osnovnih algoritamskih struktura.



# RAZGRANATA ALGORITAMSKA STRUKTURA

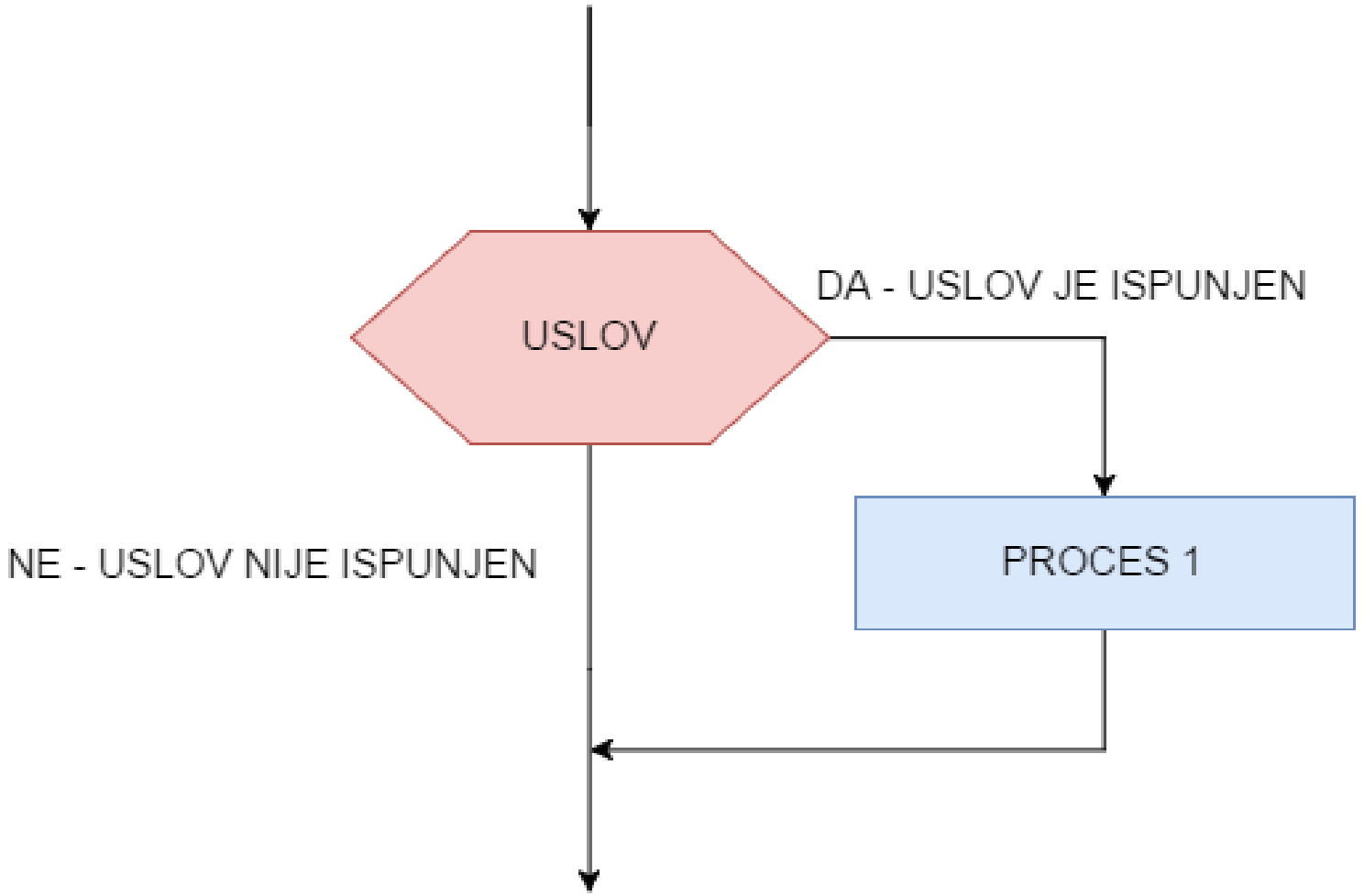
- Algoritamska struktura koja obezbeđuje izvođenje dva različita postupka pri čemu izbor zavisi od nekog uslova naziva se uslovnim grananjem ili **razgranatom algoritamskom strukturom**.

- Razgranata linijska šema je ona kod koje se svaki algoritamski korak izvršava najviše jedanput.

- To znači da postoje i algoritamski koraci koji se nikada ne izvrše.

- Ovde mora postojati bar jedan uslovni korak koji omogućava grananje algoritma.

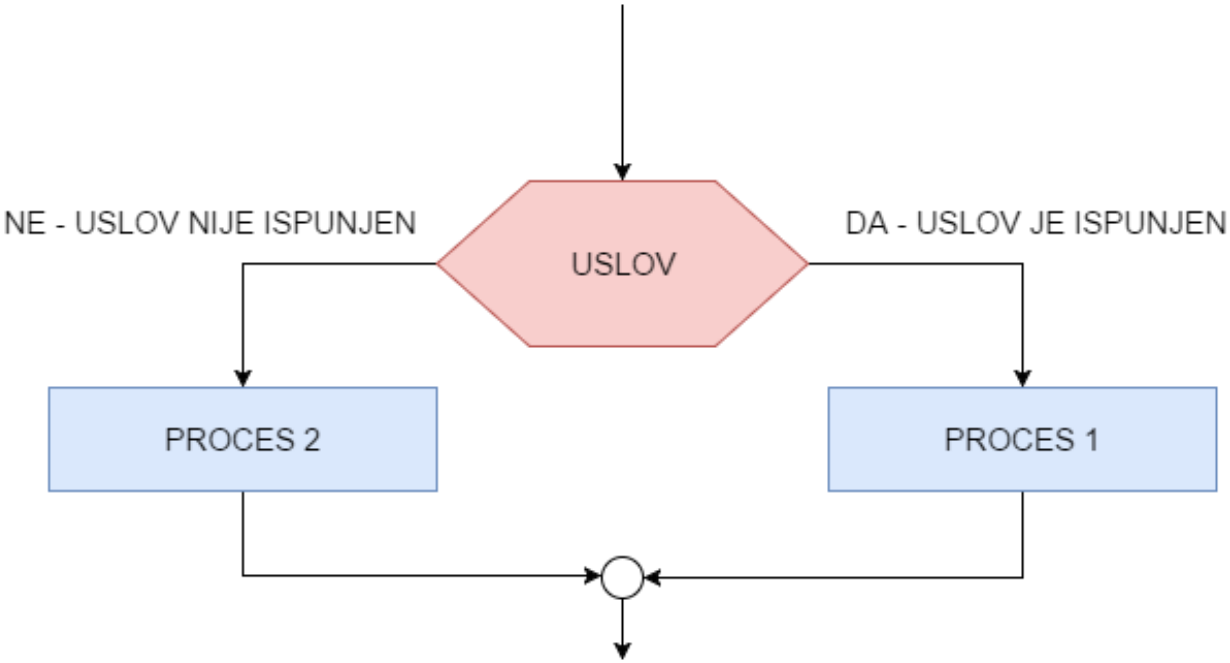
# RAZGRANATA ALGORITAMSKA STRUKTURA
# IF – THEN



# RAZGRANATA ALGORITAMSKA STRUKTURA
## IF – THEN – ELSE

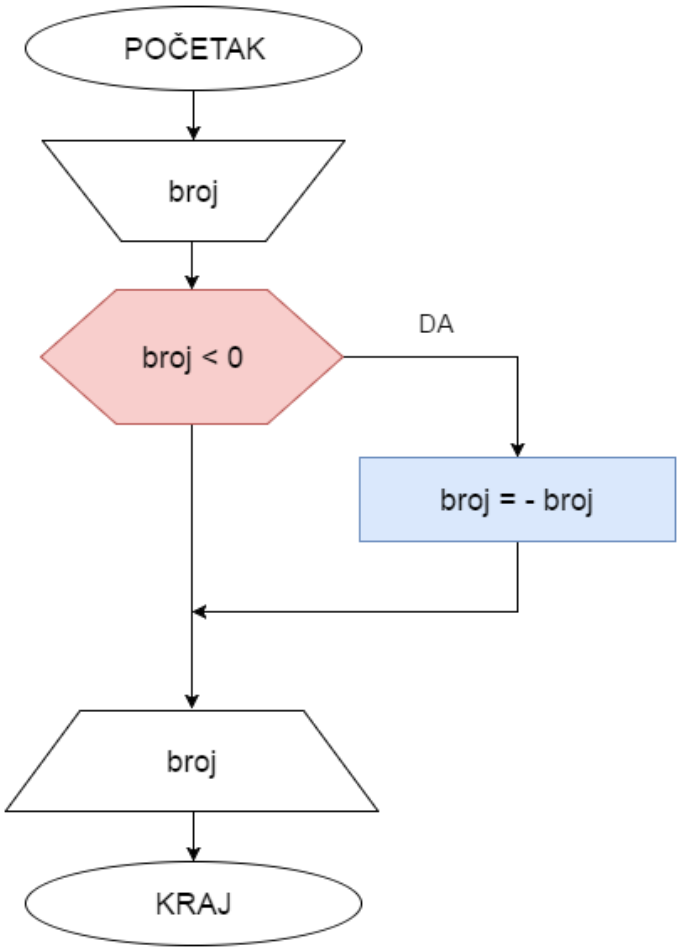
- Uslovno grananje ima tačno jednu ulaznu tačku, tačno jednu izlaznu tačku i takav tok da se svaki njen korak izvršava najviše jednom.

- Svaki korak će biti izvršen pod nekim odgovarajućim uslovom (ne postoji korak koji se ni pod kojim uslovima neće izvršiti).



# IF – THEN
## PRIMER

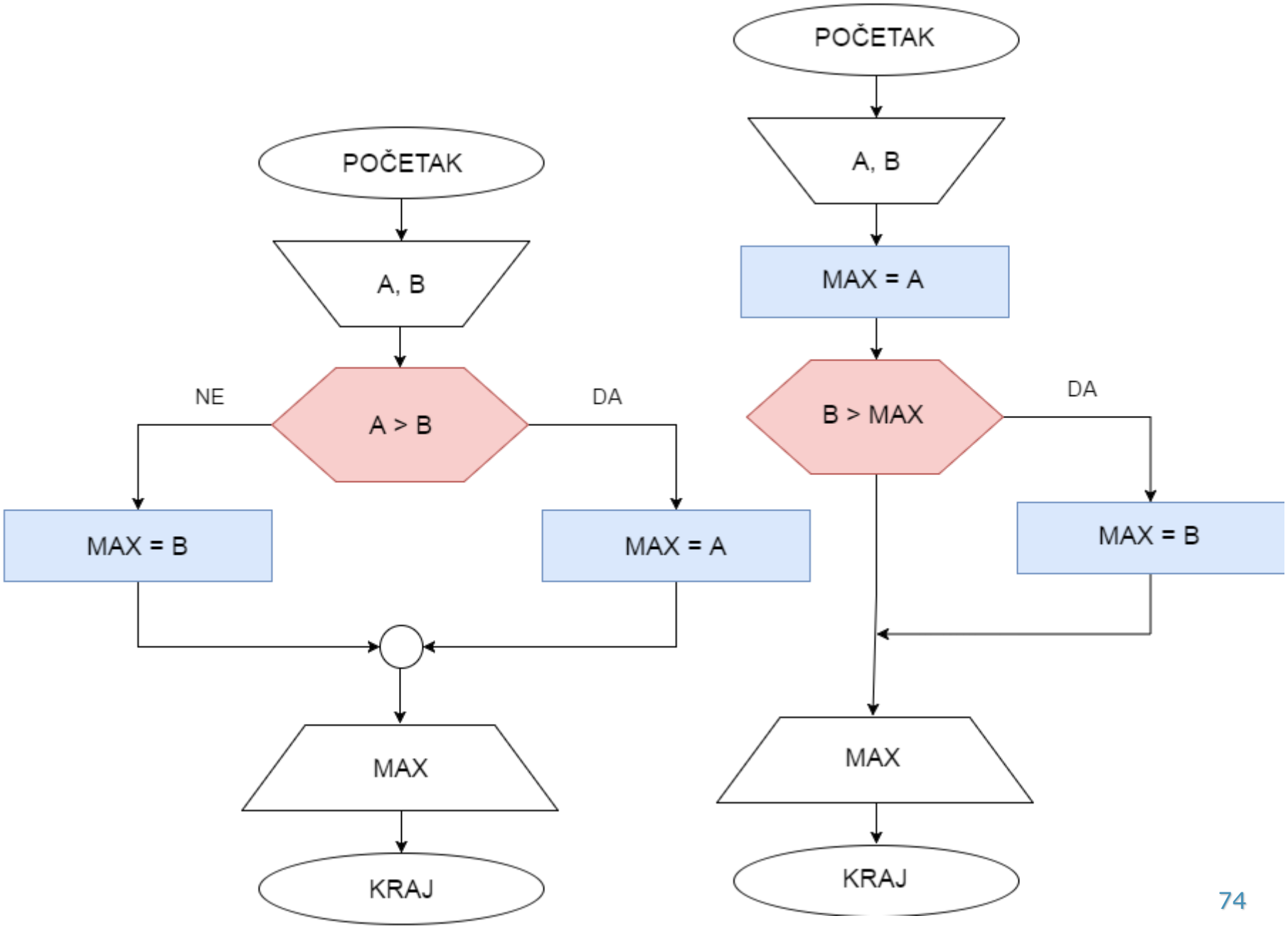
- Nacrtati dijagram toka algoritama kojim se odredjuje apsolutna vrednost broja.



# IF – THEN – ELSE
## PRIMER

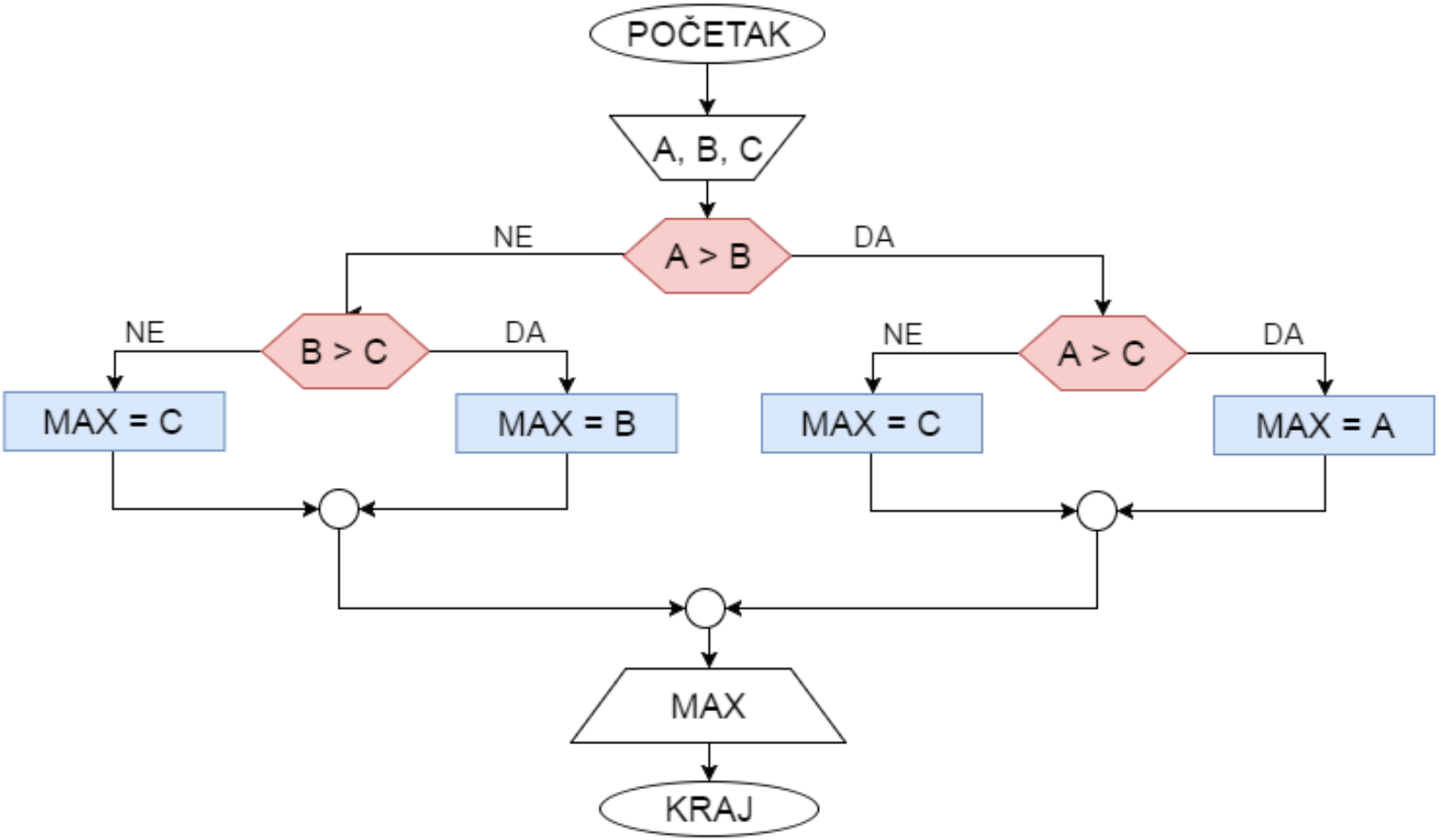
- Nacrtati dijagram toka algoritama kojim se odredjuje veći od dva zadata broja korišćenjem formule

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


POČETAK
A, B
A > B
NE
DA
MAX = B
MAX = A
MAX
KRAJ
POČETAK
A, B
MAX = A
B > MAX
DA
MAX = B
MAX
KRAJ

# IF – THEN – ELSE
## PRIMER

- Naći maksimum od tri zadata broja a, b i c $\max\{a,b,c\} = \max\{\max\{a,b\},c\}$

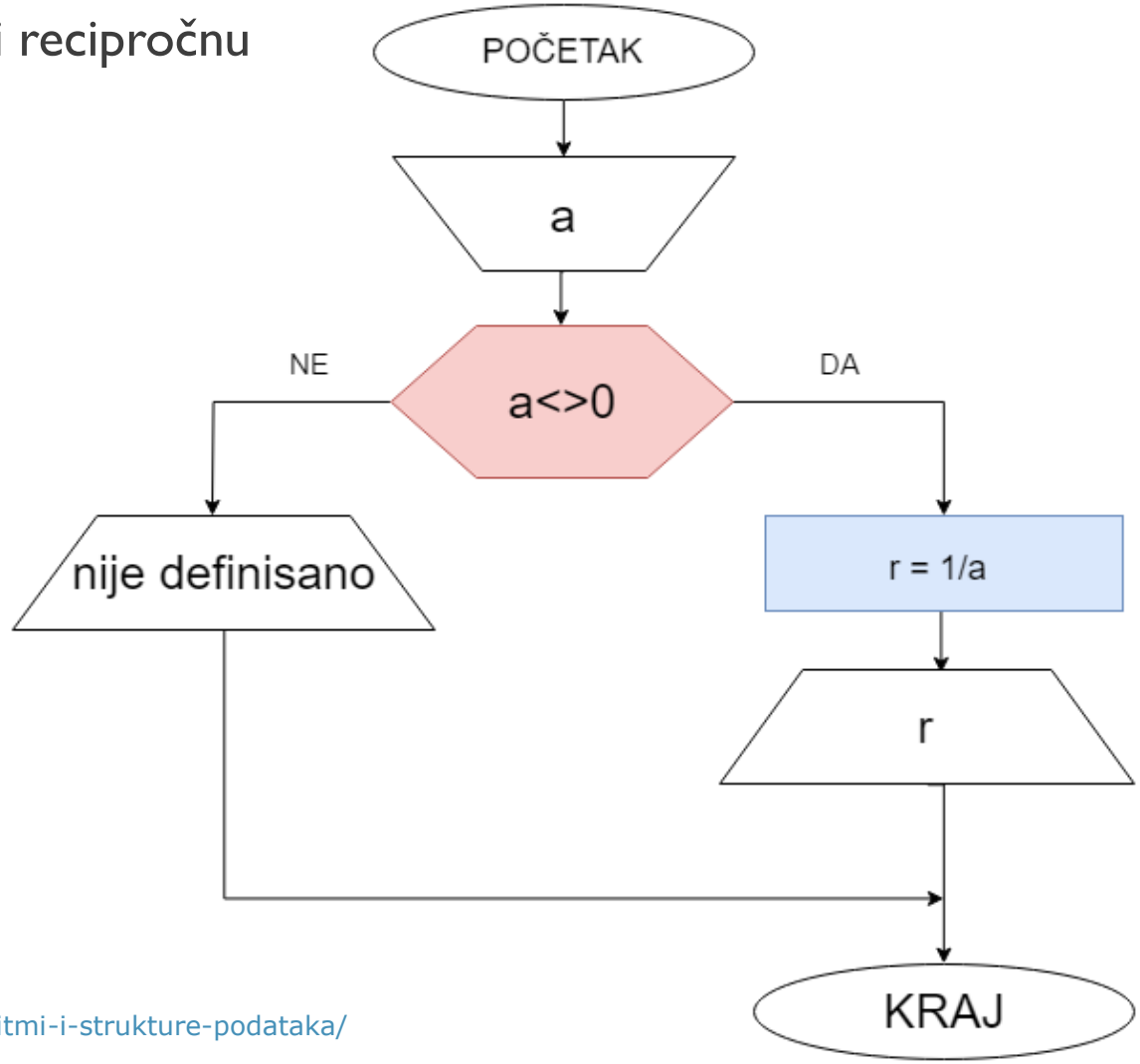


# RAZGRANATA ALGORITAMSKA STRUKURA PRIMERI

- PRIMER 1: Odrediti recipročnu vrednost broja.
- PRIMER 2: Učitati dva realna broja, ako je prvi veći ili jednak drugom, napisati njihov zbir a u suprotnom napisati njihovu razliku.
- PRIMER 3: Učitati prirodan broj. Ako je neparan ispisati njegovu recipročnu vrednost, a ako je paran ispisati recipročnu vrednost njegovog sledbenika.

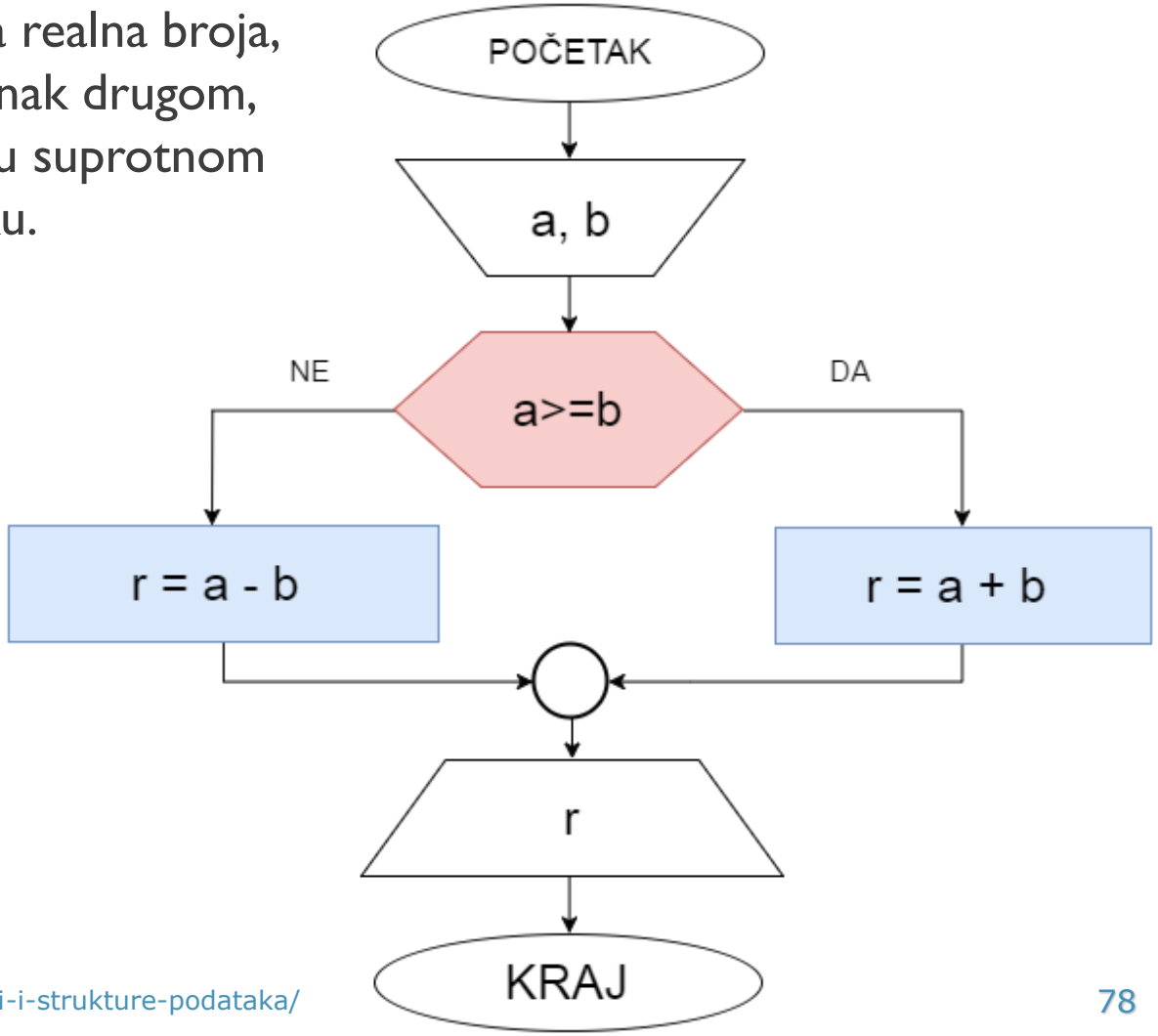
# RAZGRANATA ALGORITAMSKA STRUKURA PRIMER I REŠENJE

- PRIMER 1: Odrediti recipročnu vrednost broja.



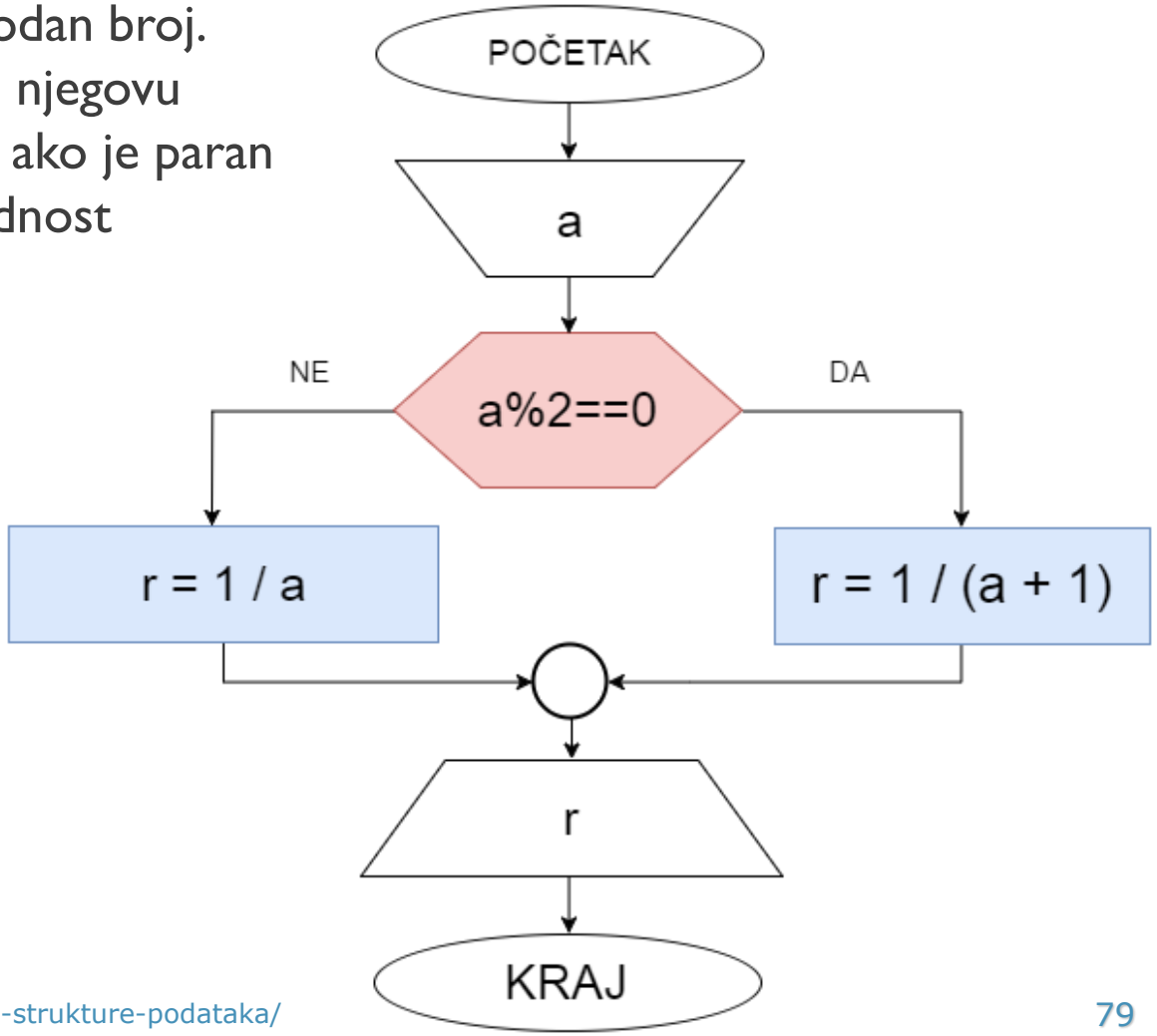
# RAZGRANATA ALGORITAMSKA STRUKURA PRIMER2 REŠENJE

- PRIMER 2: Učitati dva realna broja, ako je prvi veći ili jednak drugom, napisati njihov zbir a u suprotnom napisati njihovu razliku.



# RAZGRANATA ALGORITAMSKA STRUKURA PRIMER3 REŠENJE

- PRIMER 3: Učitati prirodan broj. Ako je neparan ispisati njegovu recipročnu vrednost, a ako je paran ispisati recipročnu vrednost njegovog sledbenika.

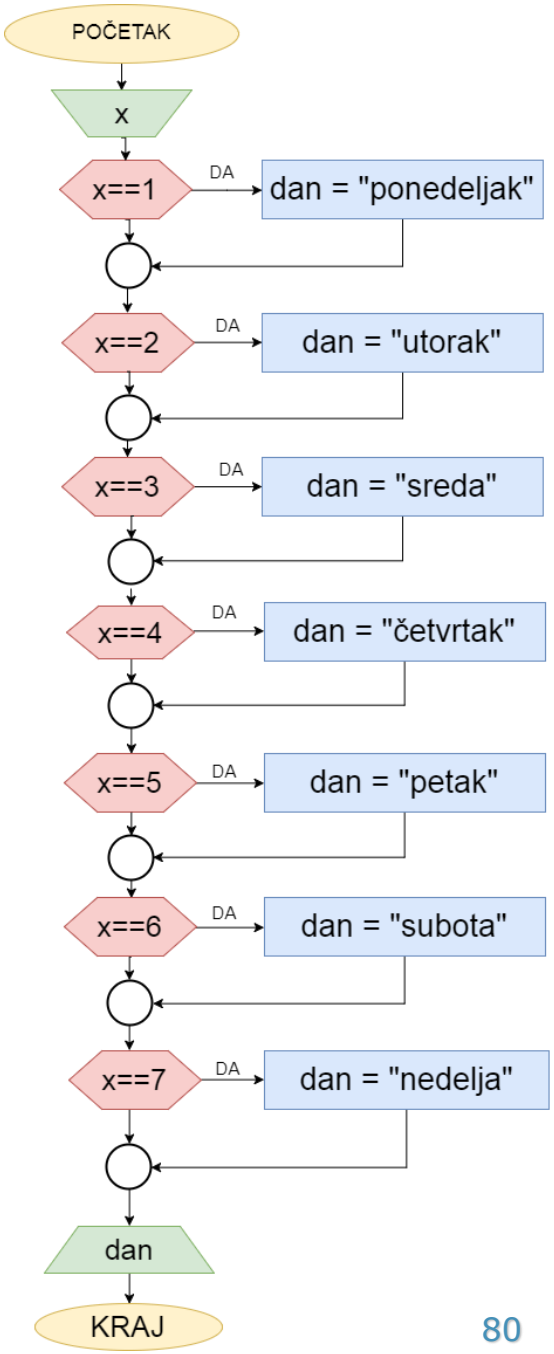


# RAZGRANATA ALGORITAMSKA STRUKTURA

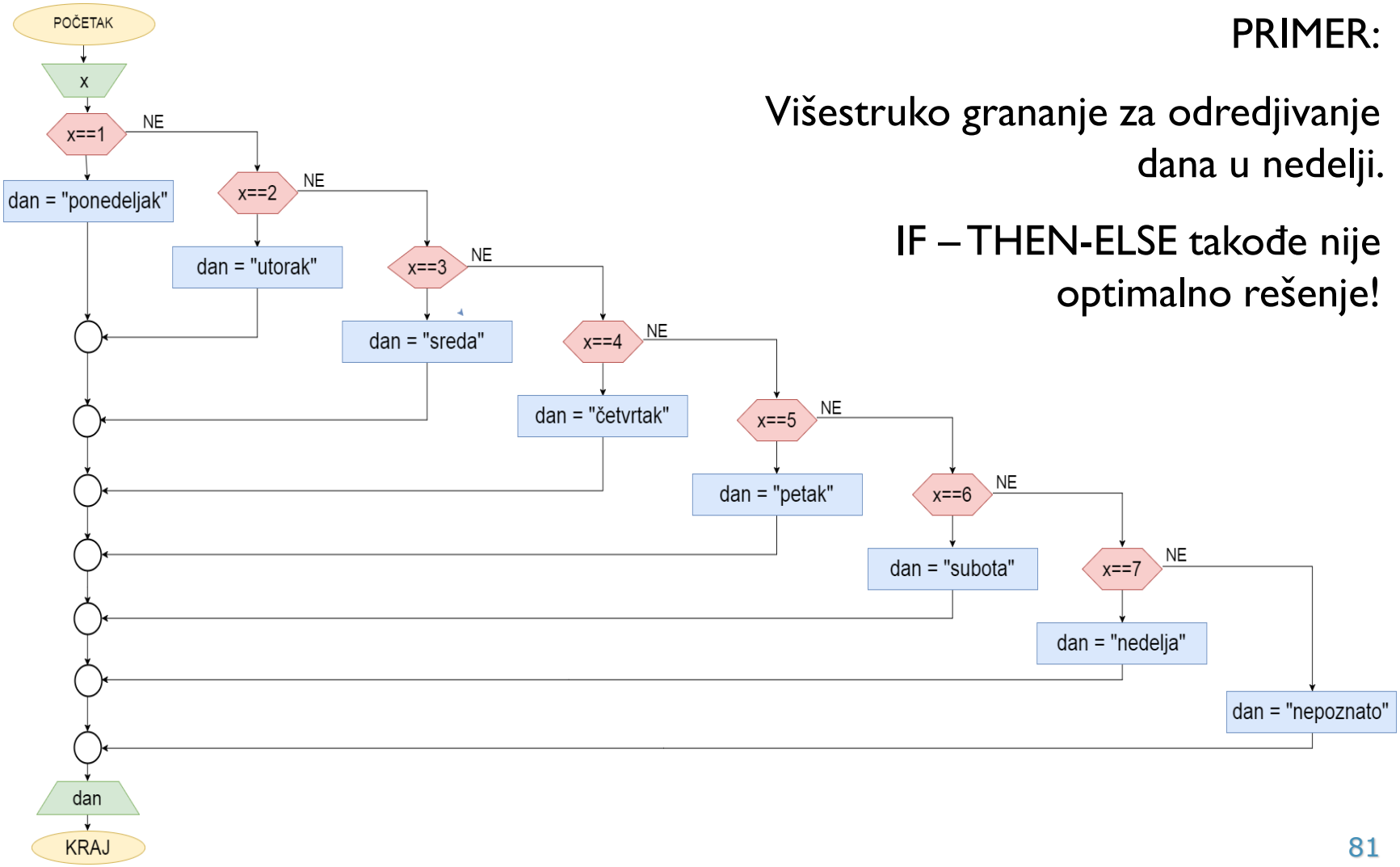
PRIMER:

Višestruko grananje za odredjivanje dana u nedelji.

IF – THEN nije optimalno rešenje!



# RAZGRANATA ALGORITAMSKA STRUKTURA



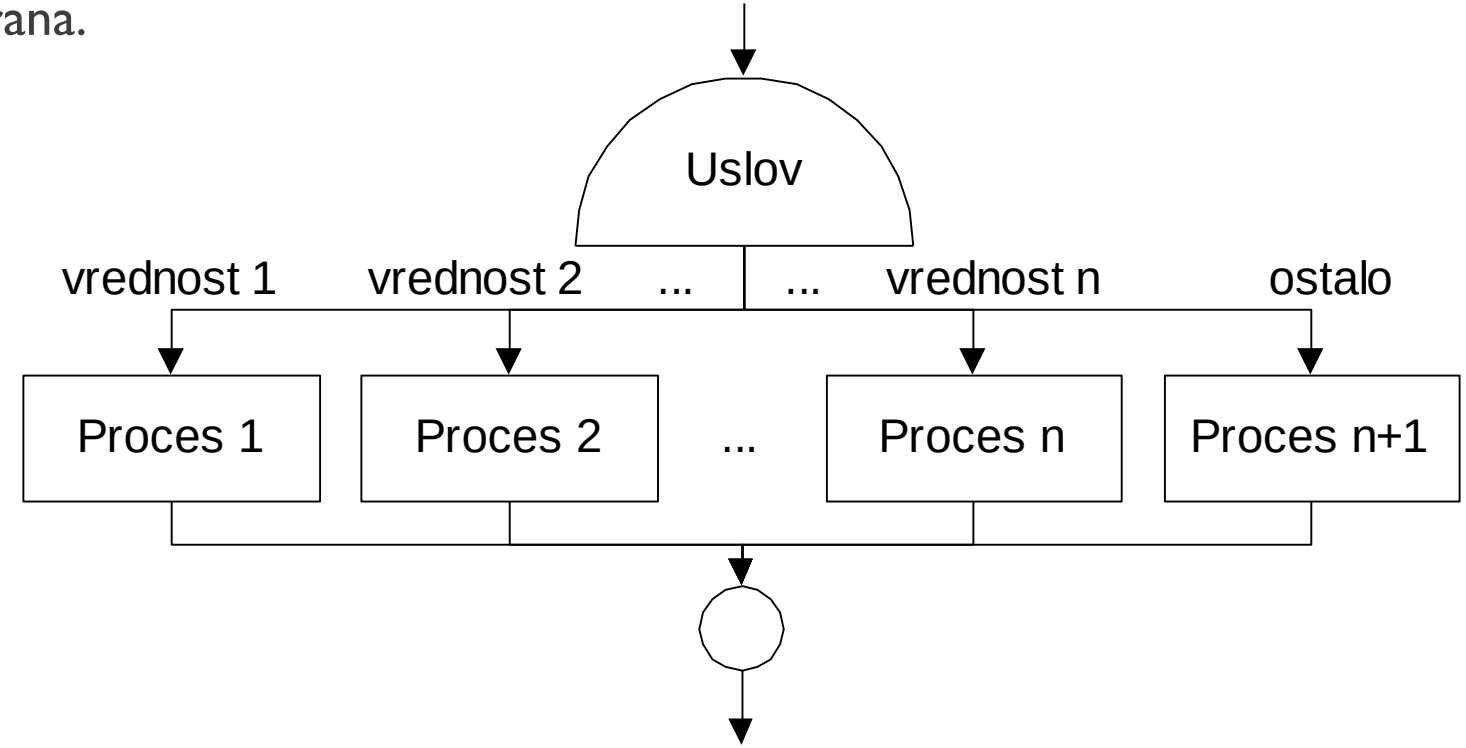

PRIMER:

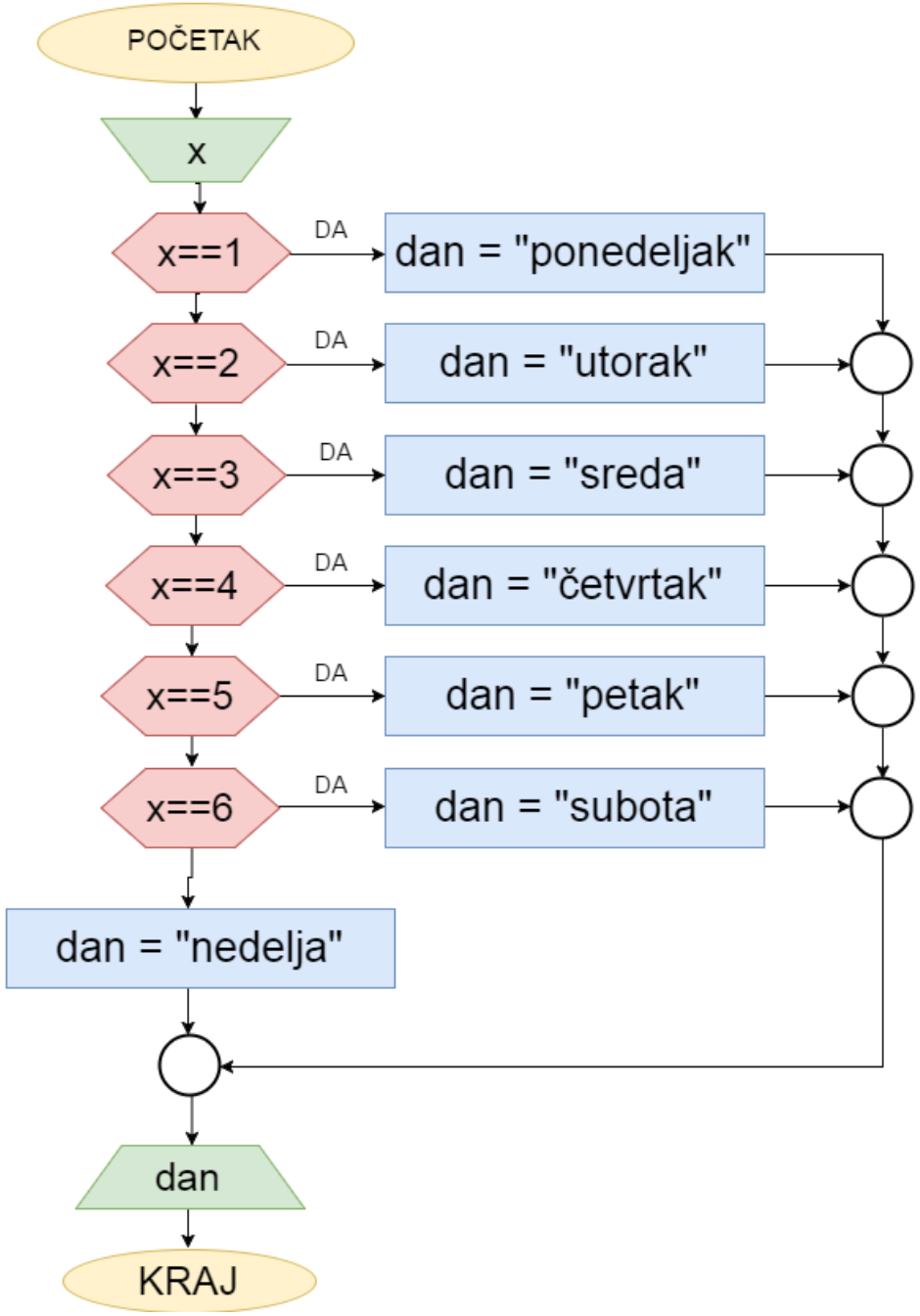
Višestruko grananje za odredjivanje dana u nedelji.

IF – THEN-ELSE takođe nije optimalno rešenje!

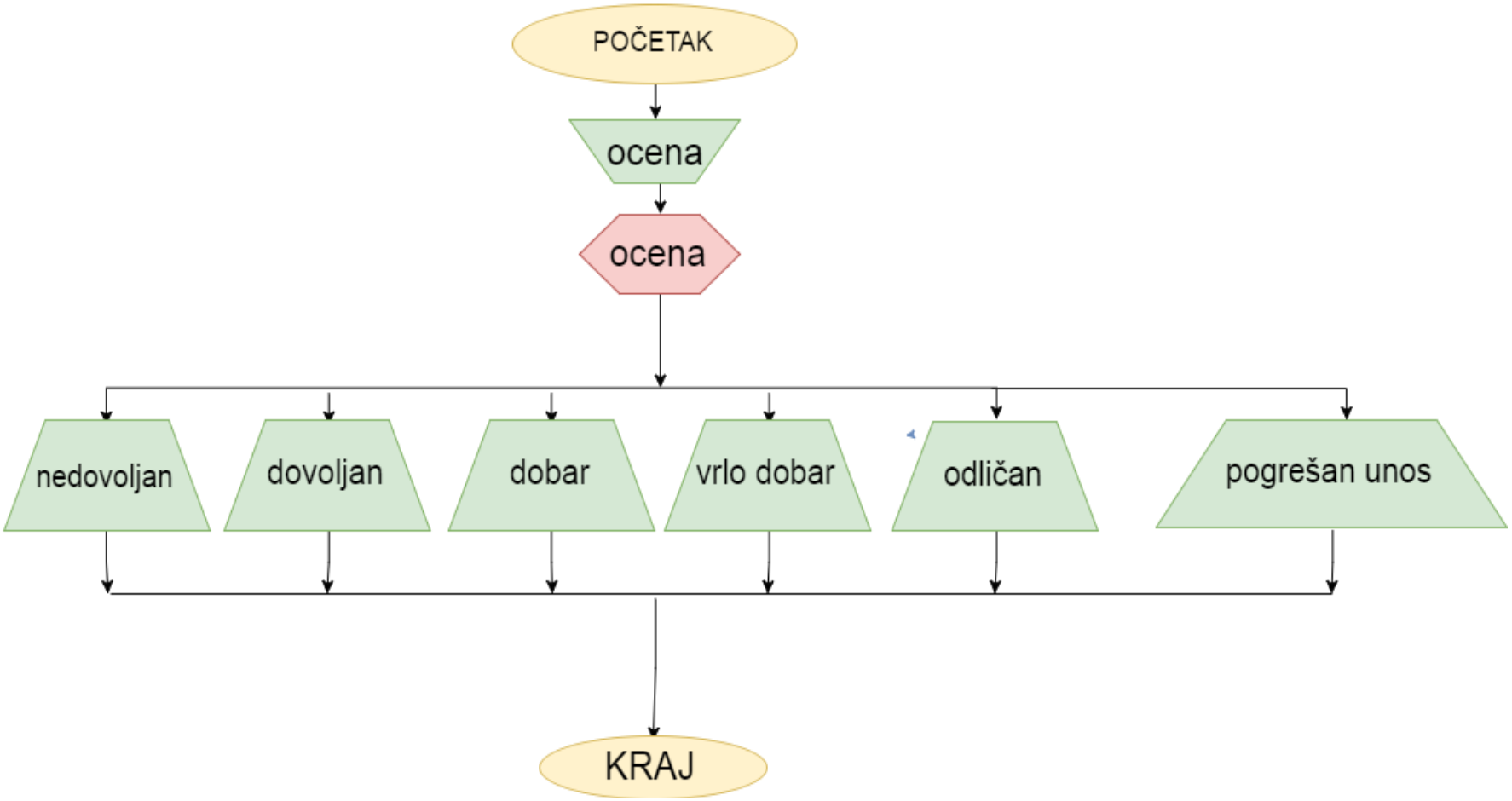
# RAZGRANATA ALGORITAMSKA STRUKTURA
# SWITCH – CASE

- Poseban slučaj uslovnog grananja je višestruko grananje.
- Višestruko grananje se može predstaviti kao više dvostrukih uslovnih grananja.
- Na osnovu vrednosti nekog izraza možemo možemo nastaviti jednom od grana.



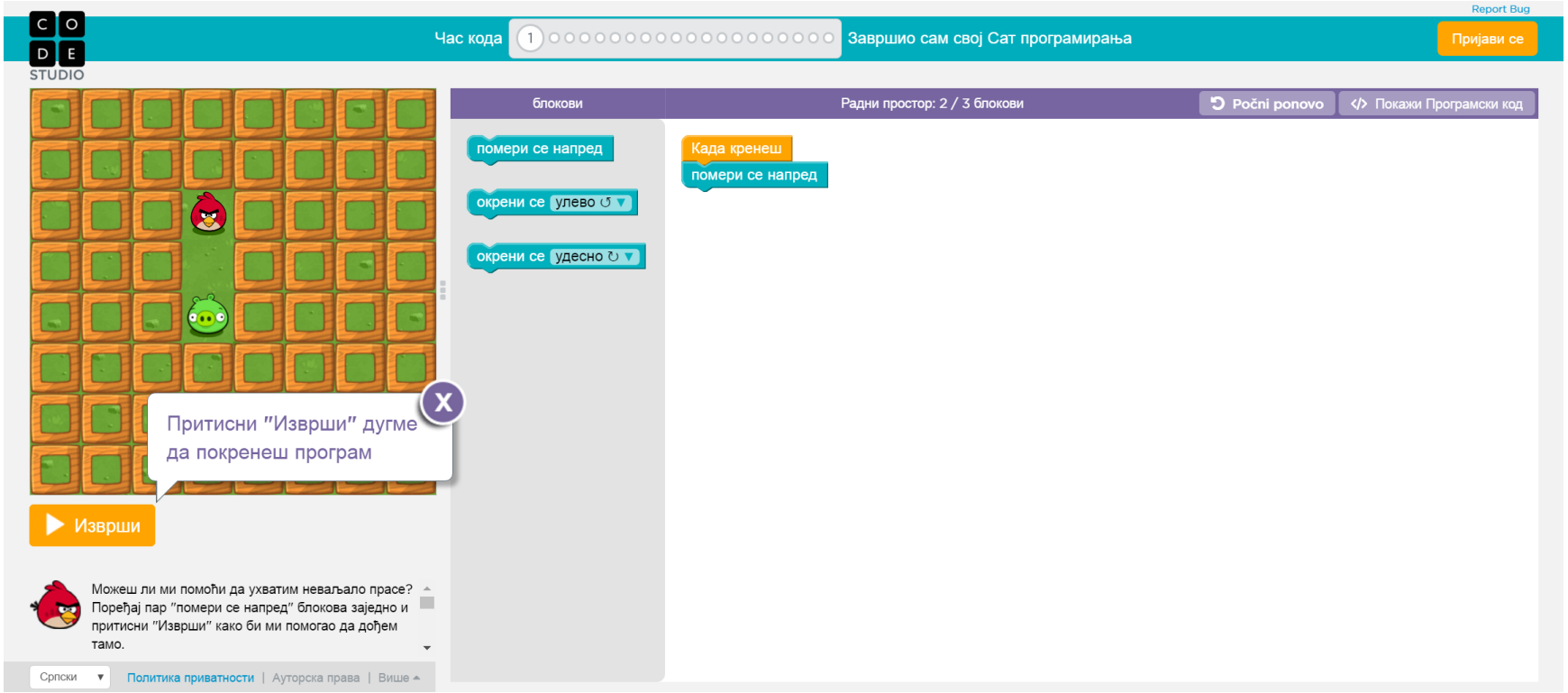

POČETAK
x
x==1
DA
dan = "ponedeljak"
x==2
DA
dan = "utorak"
x==3
DA
dan = "sreda"
x==4
DA
dan = "četvrtak"
x==5
DA
dan = "petak"
x==6
DA
dan = "subota"
dan = "nedelja"
dan
KRAJ

# RAZGRANATA ALGORITAMSKA STRUKTURA SWITCH – CASE



# ALGORITMI NA INTERNETU

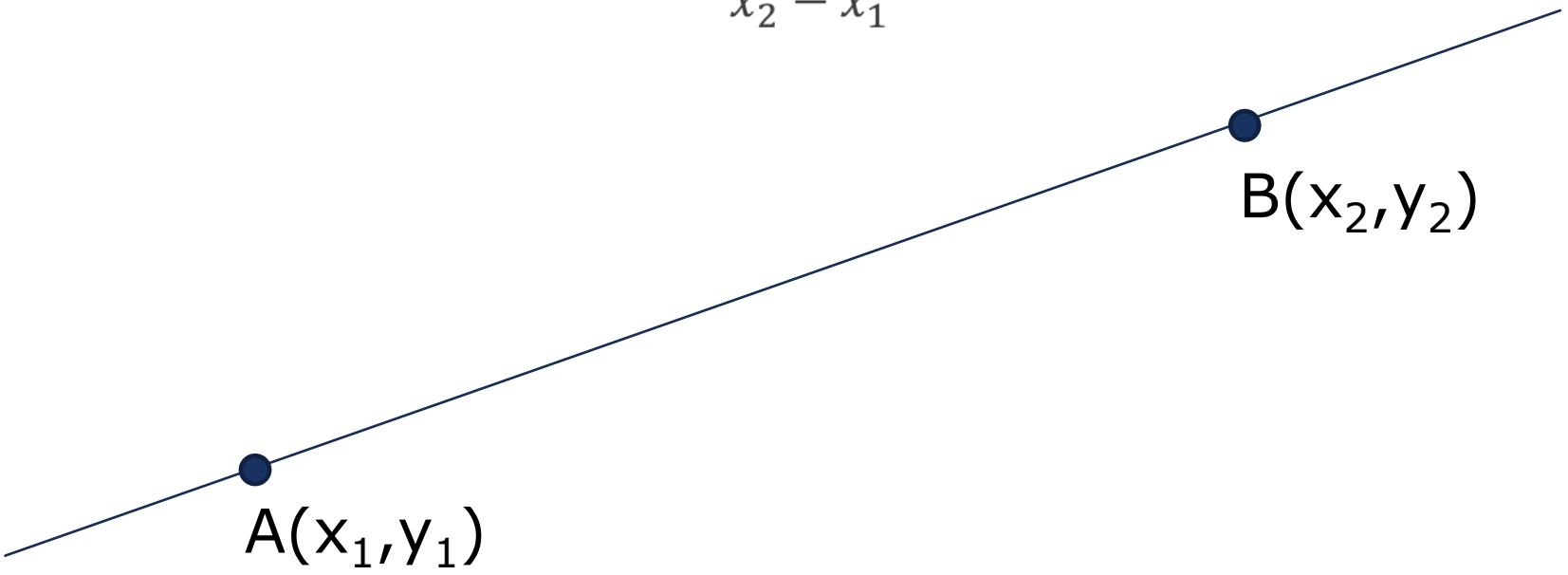
- https://studio.code.org/hoc/1



# GEOMETRIJSKI PROBLEMI

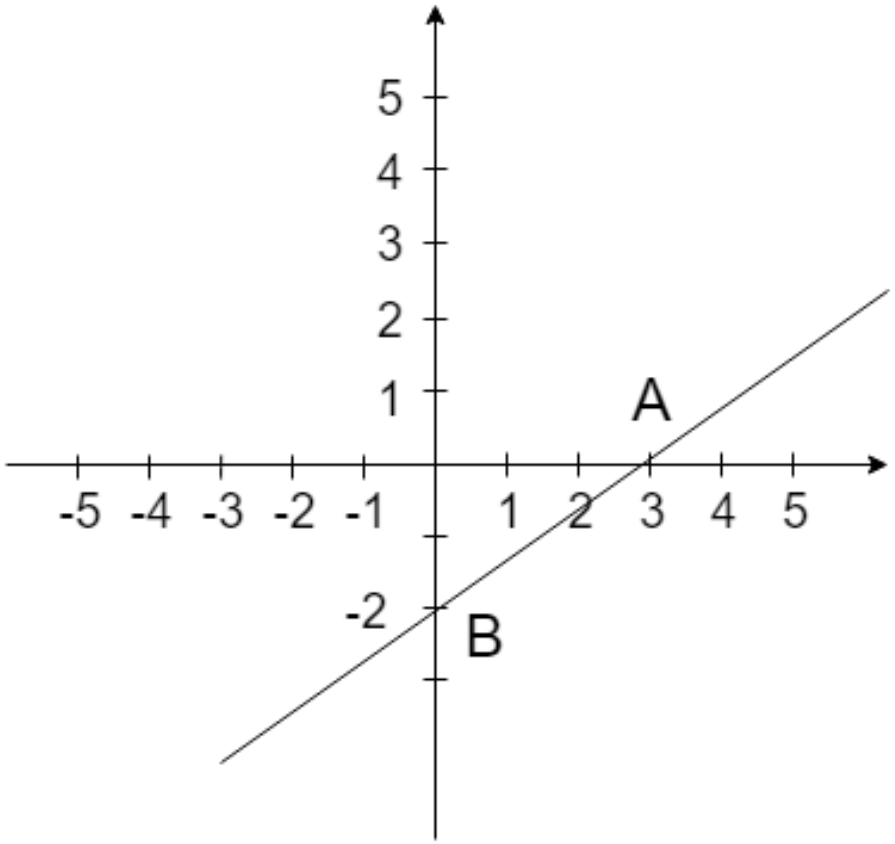
- Jednačina prave kroz dve tačke

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$



# GEOMETRIJSKI PROBLEMI
# PRIMER - PRAVA

- Nacrtati algoritam koji određuje i prikazuje sa koje strane prave se nalazi tačka p sa koordinatama x, y.



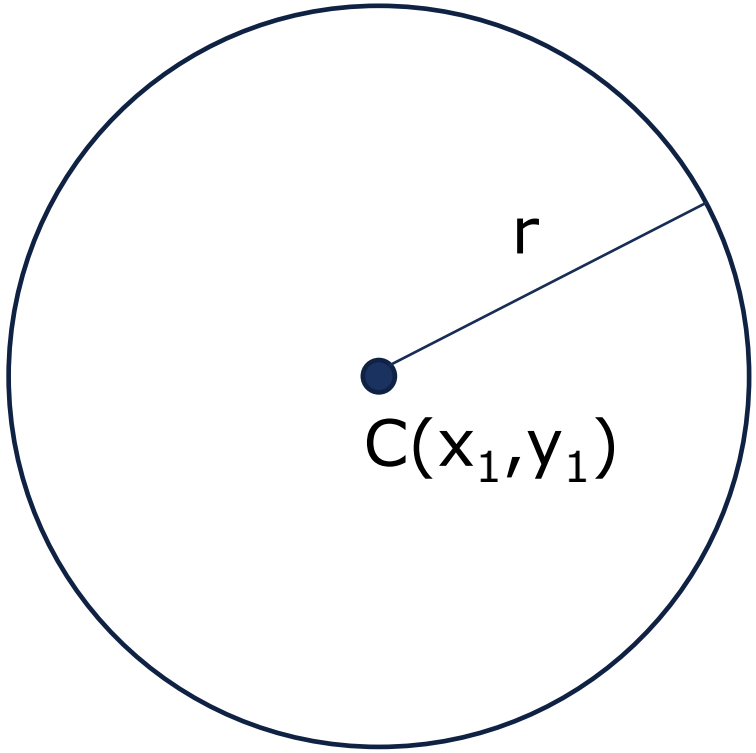

A (x1,y1) = A (3,0)

B (x2,y2) = B (0,-2)

# GEOMETRIJSKI PROBLEMI

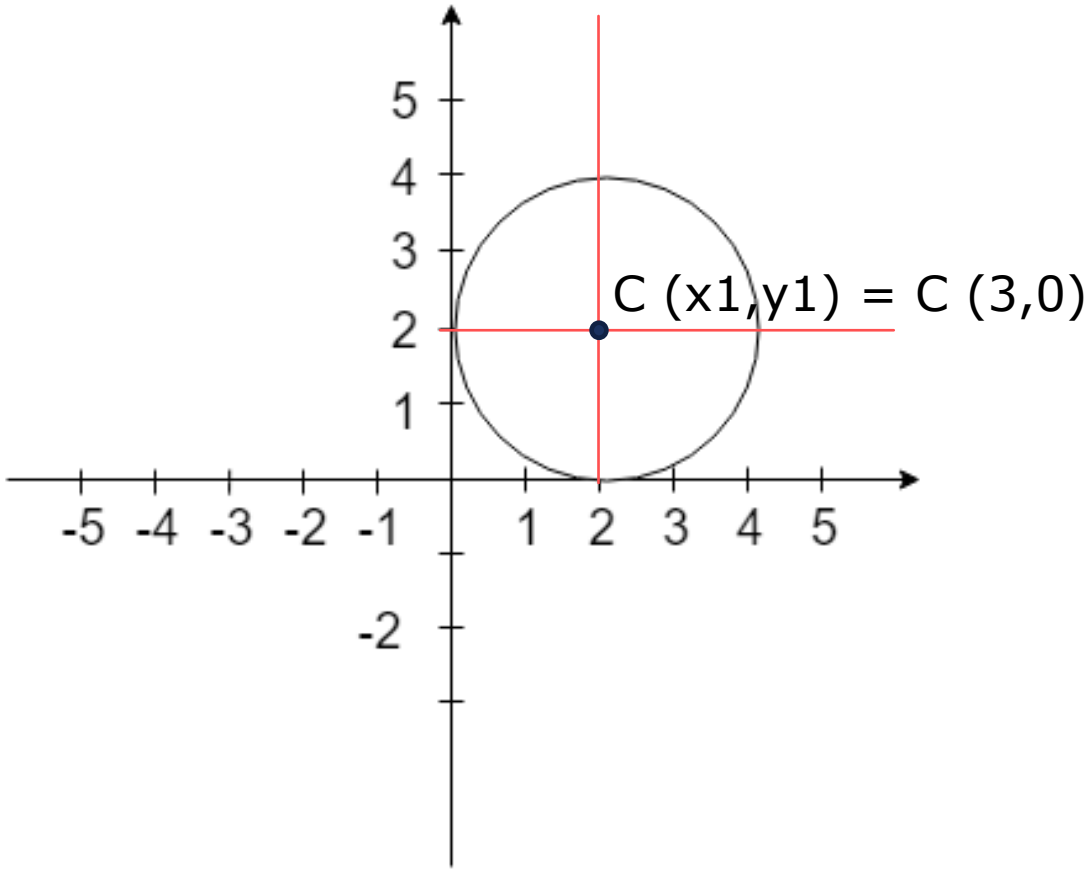
- Jednačina kružnice

$$(x - x_1)^2 + (y - y_1)^2 = r^2$$

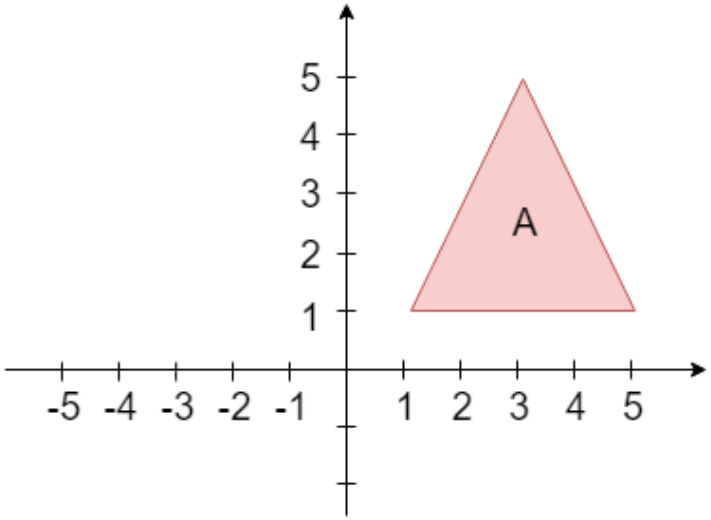


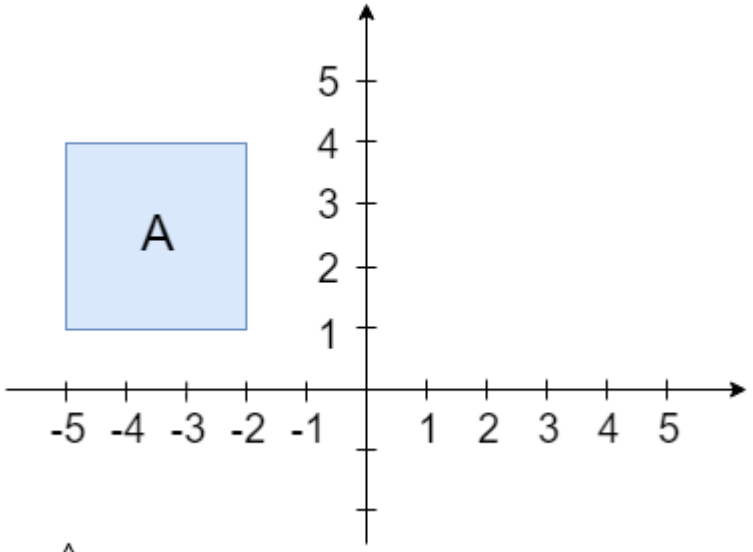
# GEOMETRIJSKI PROBLEMI
# PRIMER - PRAVA

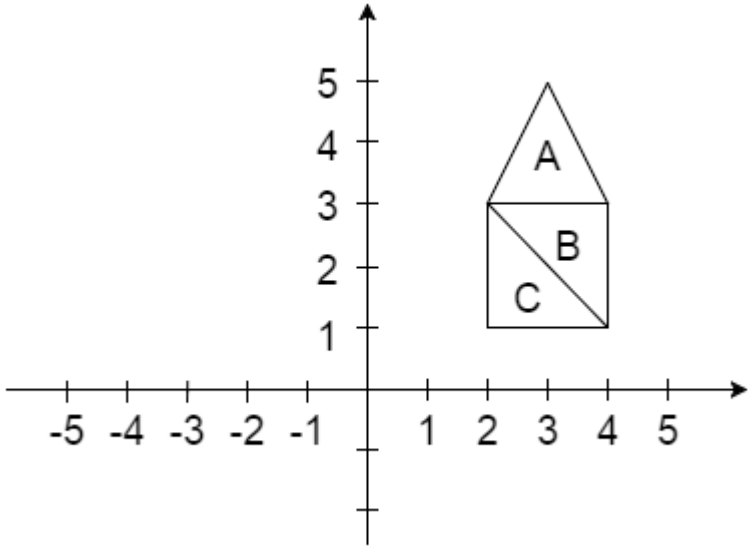
- Nacrtati algoritam koji određuje i prikazuje sa koje strane kružnice se nalazi tačka p sa koordinatama x, y.



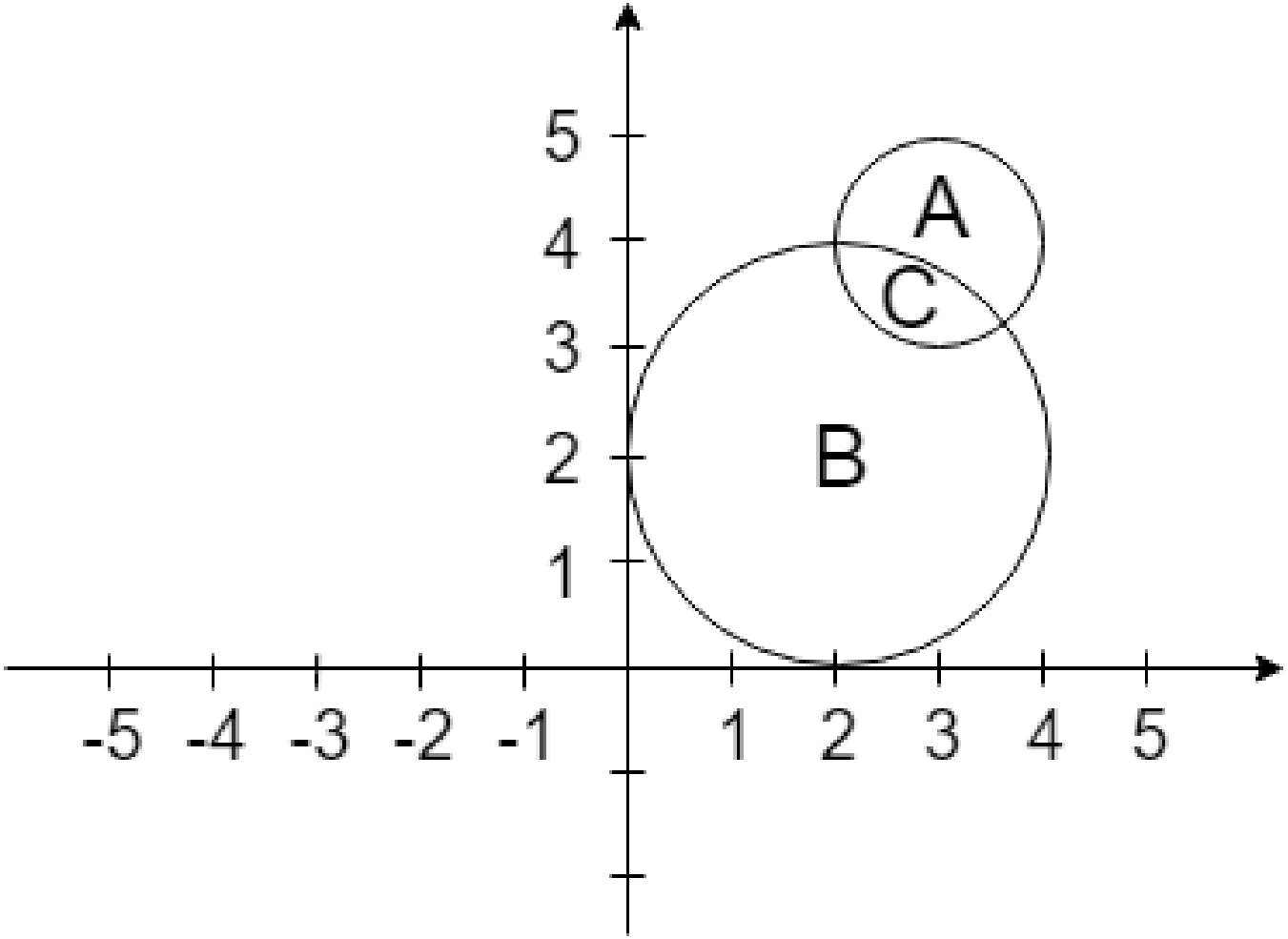
# GEOMETRIJSKI PROBLEMI
# PRIMERI



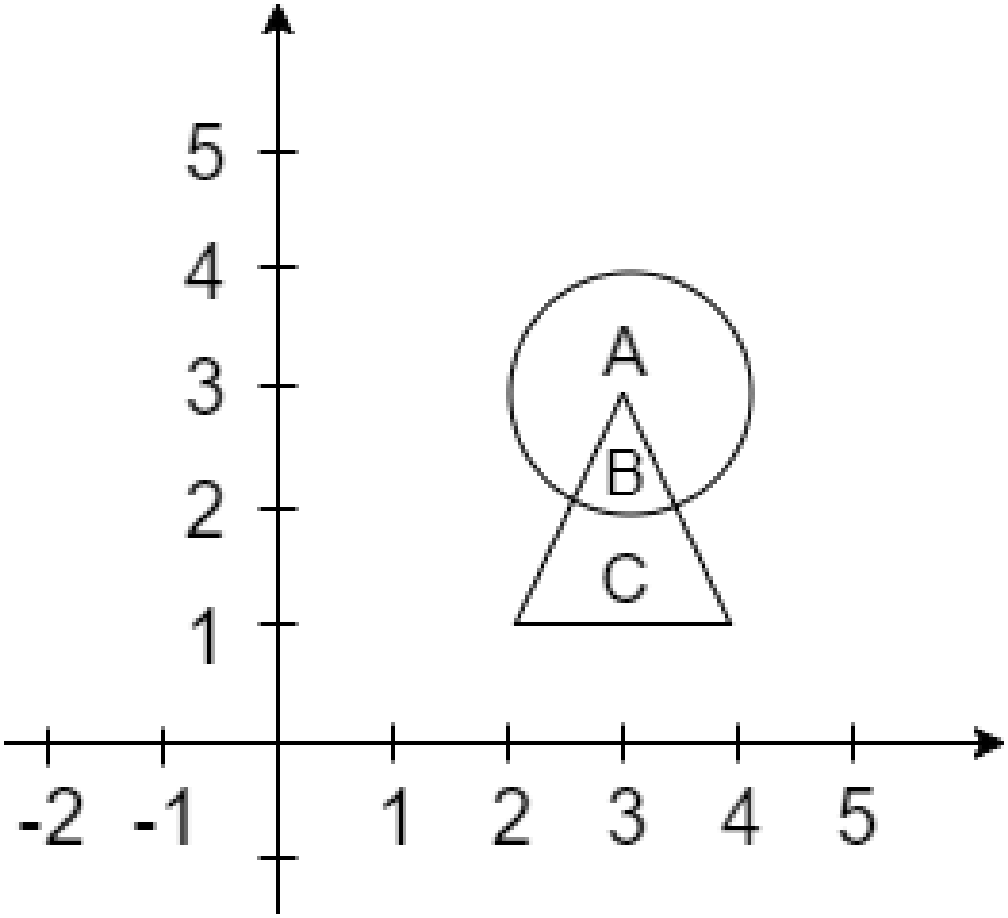



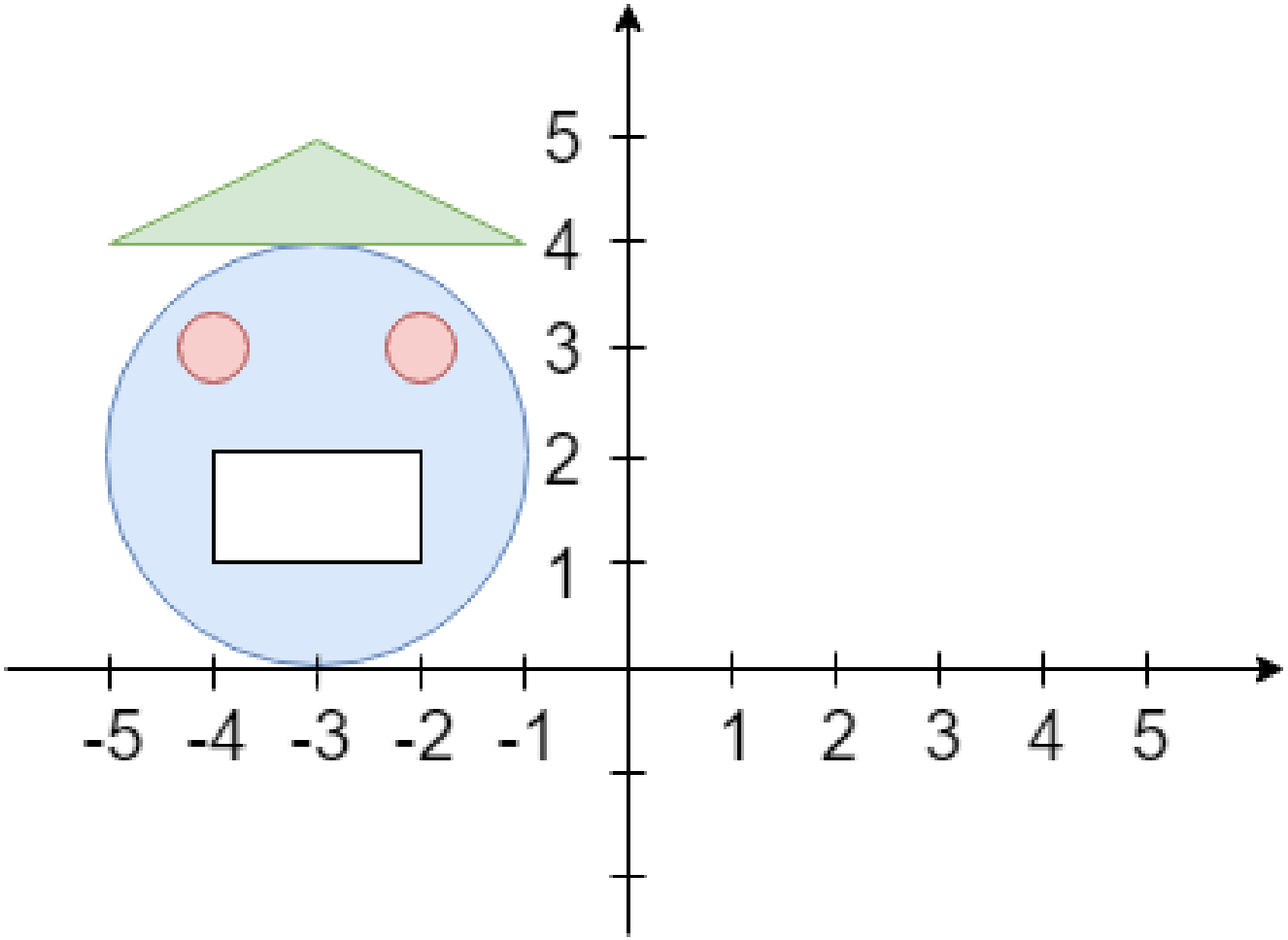
# GEOMETRIJSKI PROBLEMI
# PRIMERI



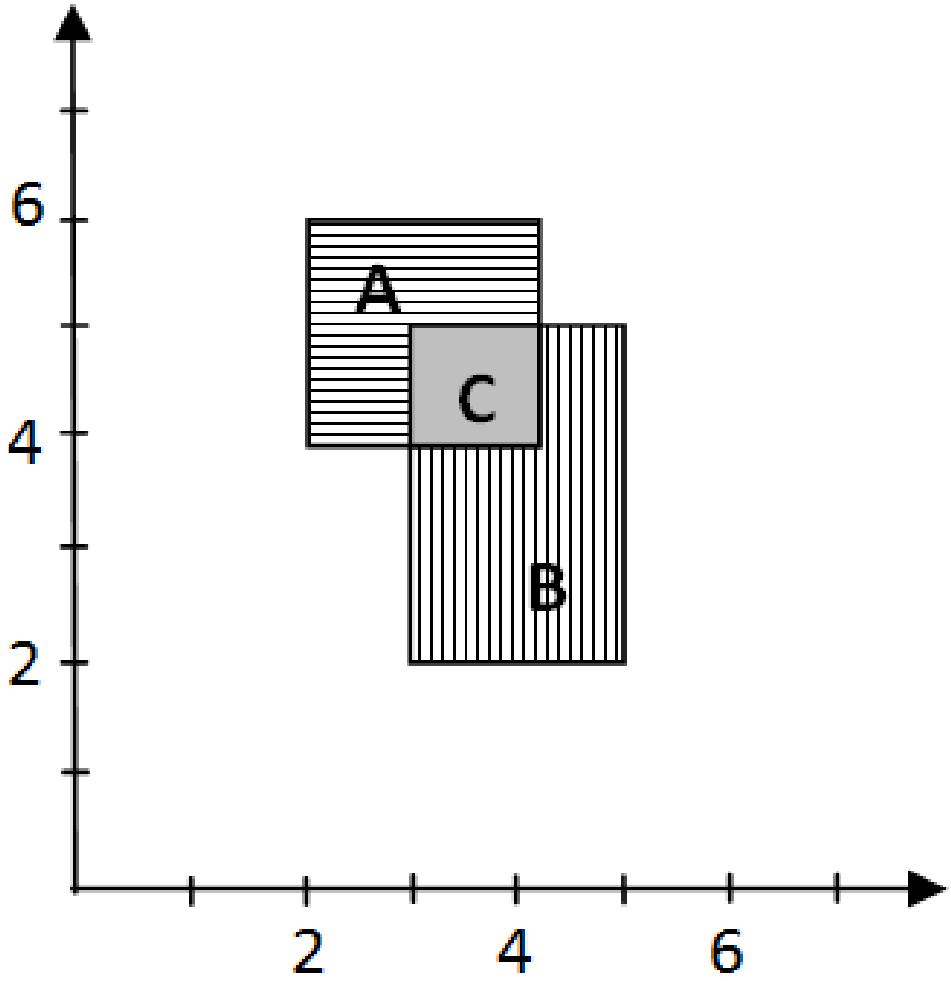
# GEOMETRIJSKI PROBLEMI
# PRIMERI



# GEOMETRIJSKI PROBLEMI
# PRIMERI



# GEOMETRIJSKI PROBLEMI
# PRIMERI



# ALGORITMI I STRUKTURE PODATAKA

- PREDAVANJE 4 -

# CIKLIČNA ALGORITAMSKA STRUKTURA

- Ponavljanjem, ili cikličnom algoritamskom strukturom nazivamo strukturu koja obezbeđuje ponavljanje nekih koraka algoritma.

- Ciklična algoritamska struktura ima:
  - tačno jednu ulaznu tačku,
  - tačno jednu izlaznu tačku
  - u zavisnosti od nekog uslova:
    - koraci koji se ponavljaju se izvršavaju
    - izlazi se iz strukture.

# CIKLIČNA ALGORITAMSKA STRUKTURA

- Ciklična struktura zove se još i petlja, a uslov za izlazak iz petlje zove se izlazni kriterijum.
- Izlazni kriterijum je najčešće ili broj izvršenih ciklusa ili dostignuta tačnost u računanju.

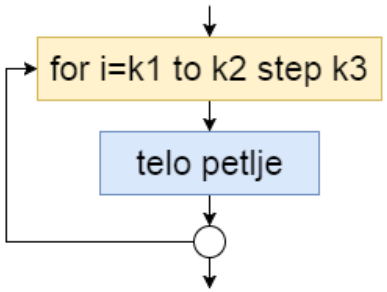
Osnovne vrste ciklusa ili petlji su sledeće:

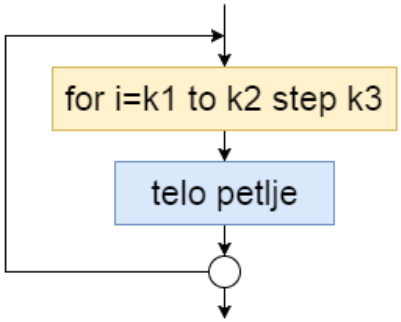
- Petlje sa eksplicitnim brojačem (FOR)
- Petlje sa uslovnim izlazom (WHILE, REPEAT – UNTIL, DO-WHILE)
- Beskonačne petlje sa i bez mehanizma uslovnog izlaza (LOOP)
- Petlje po elementima skupa (FOREACH).

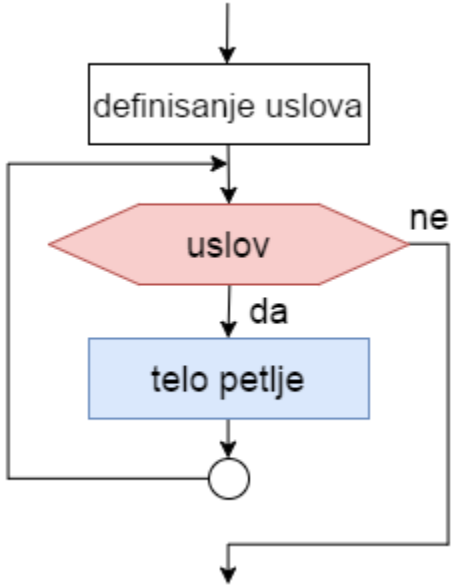
# CIKLIČNA ALGORITAMSKA STRUKTURA

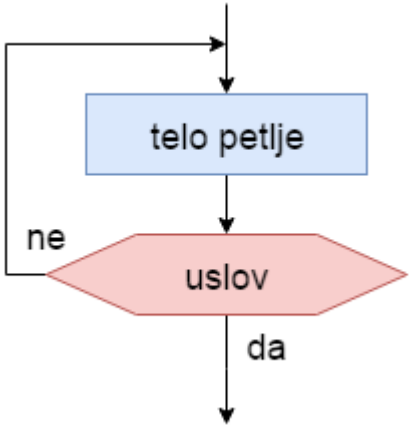
U upotrebi je više cikličnih struktura:

- brojačka struktura *FOR* – ponavljanje određeni broj puta
- WHILE struktura – nije unapred definisan broj ponavljanja
- REPEAT-UNTIL struktura – nije unapred definisan broj ponavljanja
- DO-WHILE struktura – isto što i REPEAT-UNTIL



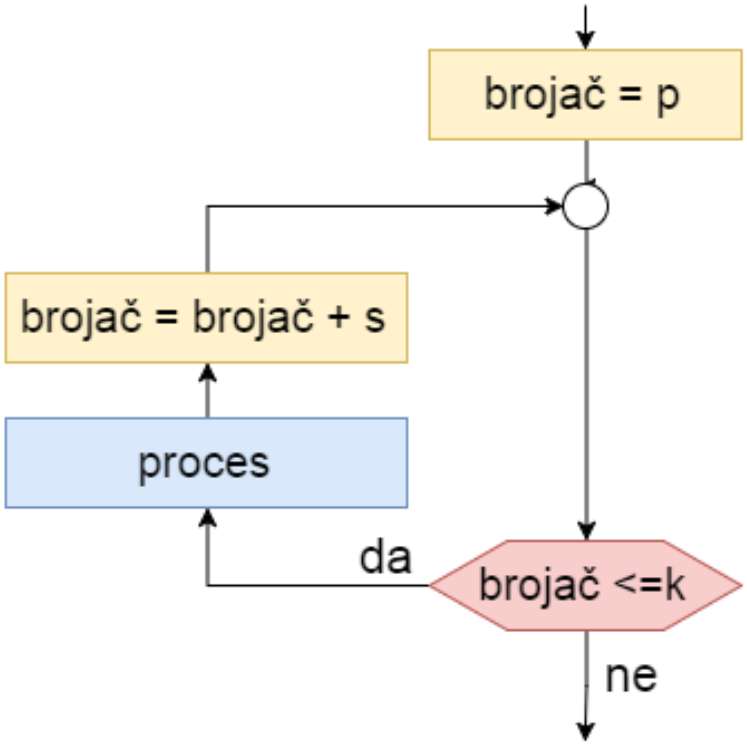



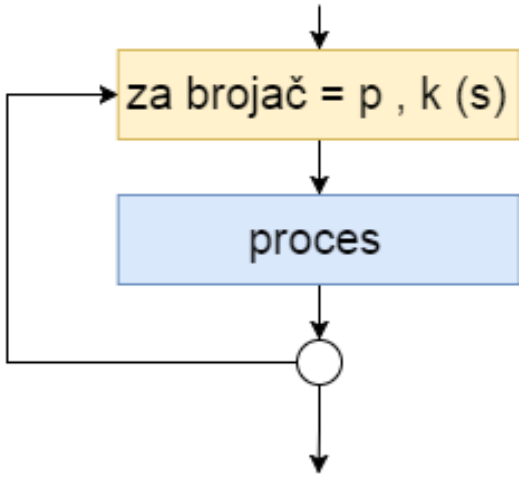




brojačka petlja | while | repeat-until

# BROJAČKA CIKLIČNA STRUKTURA - FOR

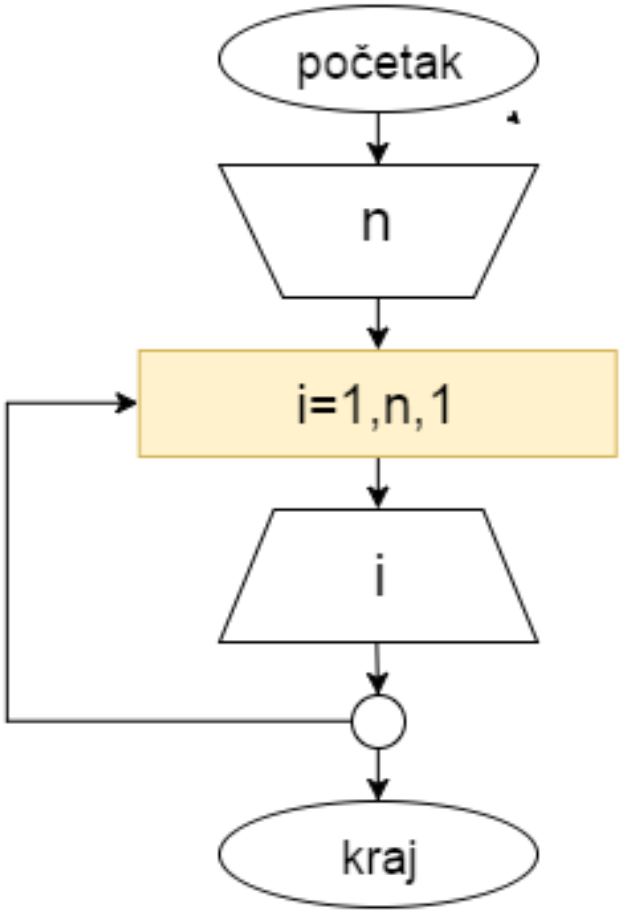
- Brojačka struktura obezbeđuje ponavljanje nekog procesa zadati broj puta.
- Ulazna tačka je početak brojanja, izlazna tačka je kraj brojanja.
- Brojačka struktura predstavlja skraćivanje zapisa WHILE strukture (s je korak, k je zadati uslov)



# PRIMER
# BROJAČKA CIKLIČNA STRUKTURA - FOR

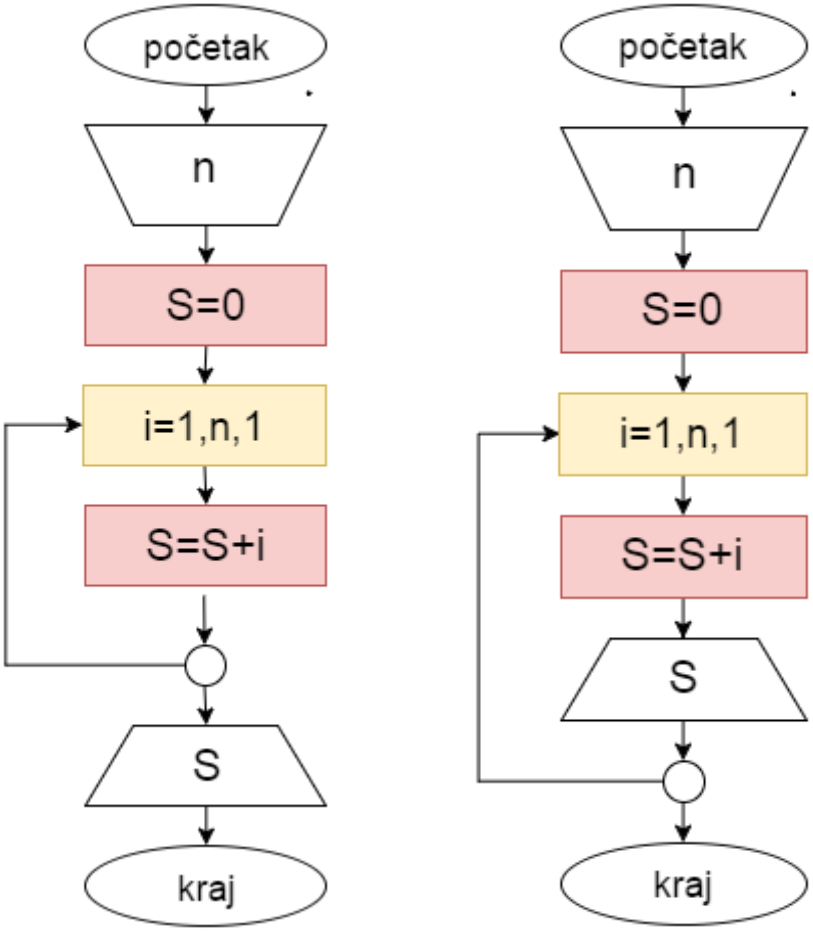
**Primer.** Ispisati prvih n prirodnih brojeva ( FOR petlja).



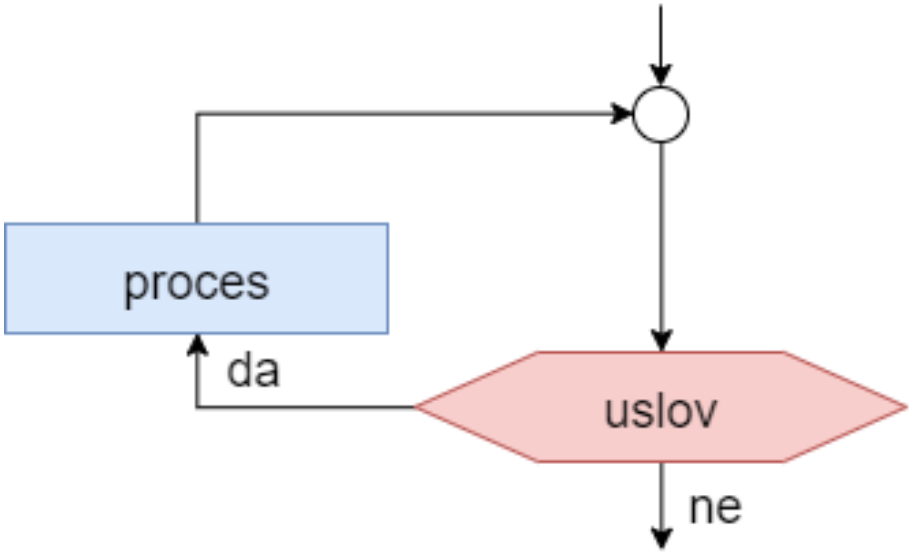

1. **Učitati do kog broja (n)**
2. **Za i=1 do n radi**
3. **Ispiši vrednost i**
4. **Idi na sledeći korak**

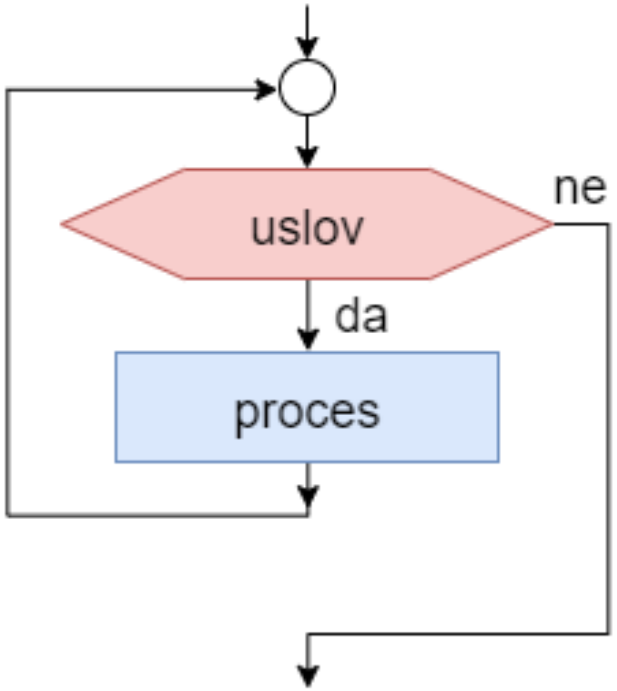
# PRIMER
# BROJAČKA CIKLIČNA STRUKTURA - FOR

- Sabrati prvih n prirodnih brojeva.

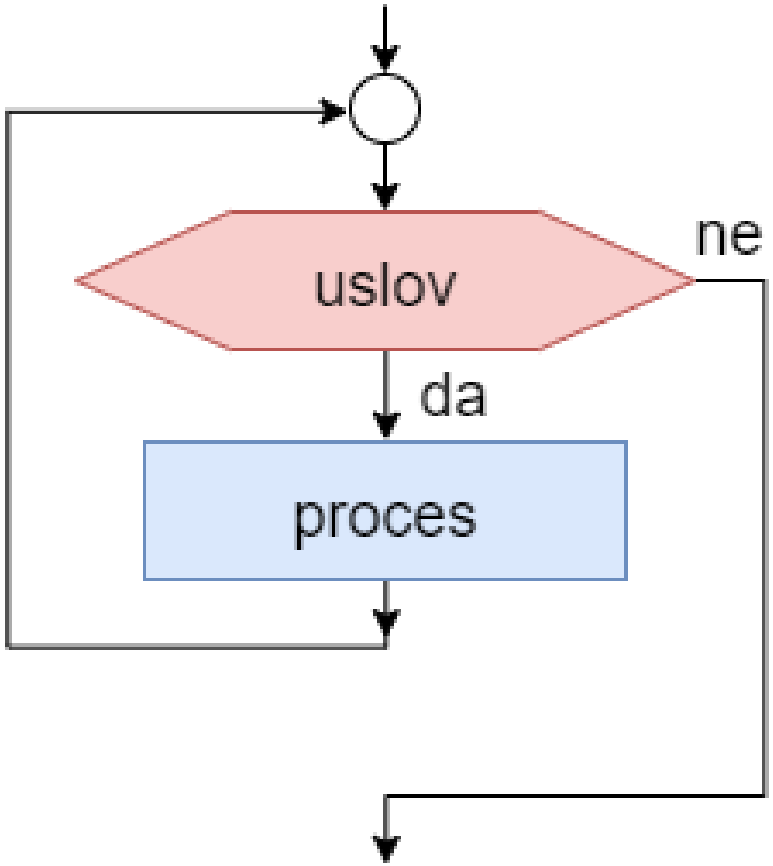


# DO-WHILE CIKLIČNA STRUKTURA



# DO-WHILE CIKLIČNA STRUKTURA

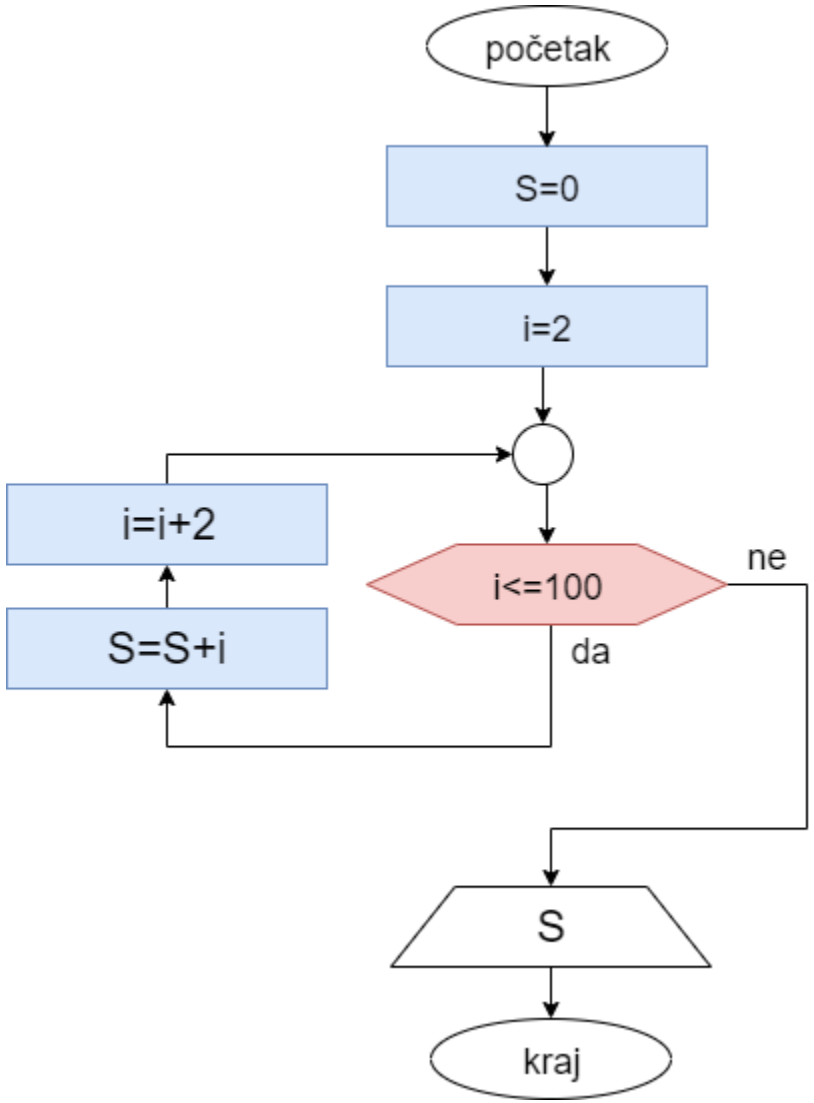



Tok ciklusa petlje sledi:

1. Ispitaj uslov
2. Ako je uslov petlje ispunjen izvrši proces
3. Ako uslov petlje nije ispunjen idi na kraj (ili na sledeći korak)

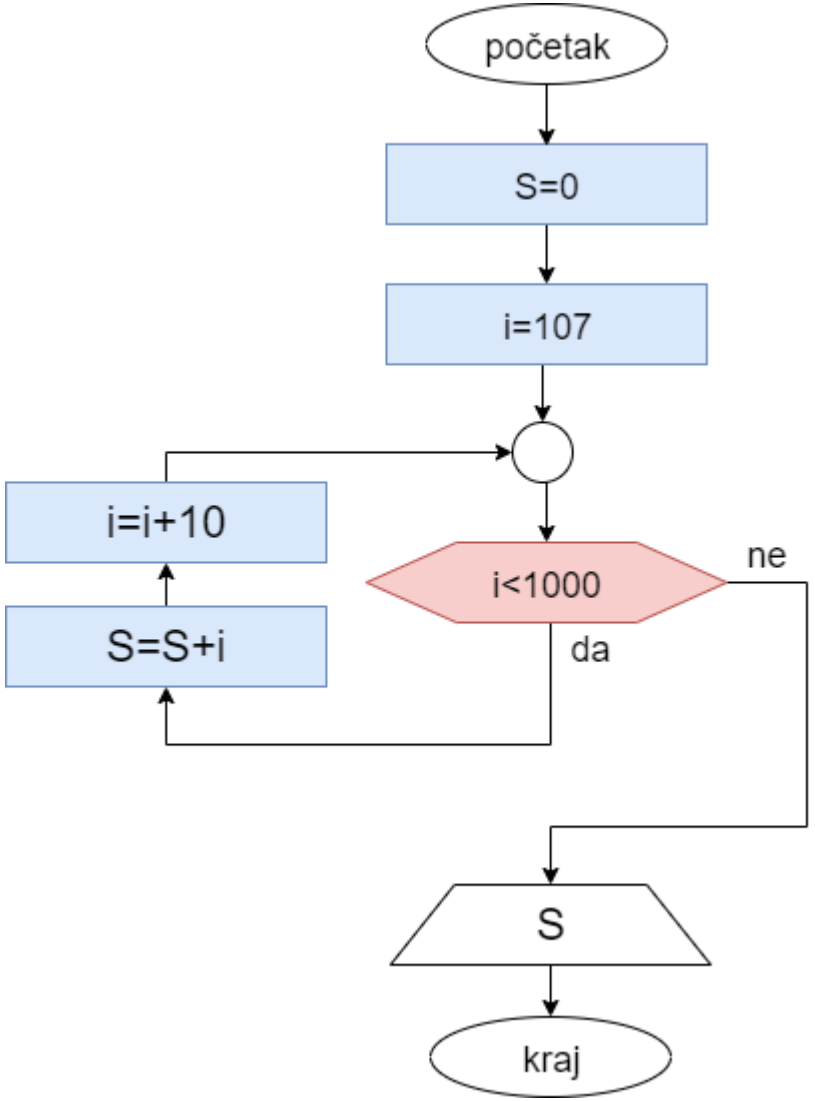
# PRIMER
# DO-WHILE CIKLIČNA STRUKTURA

- Sabrati parne brojeve od 1 do 100.



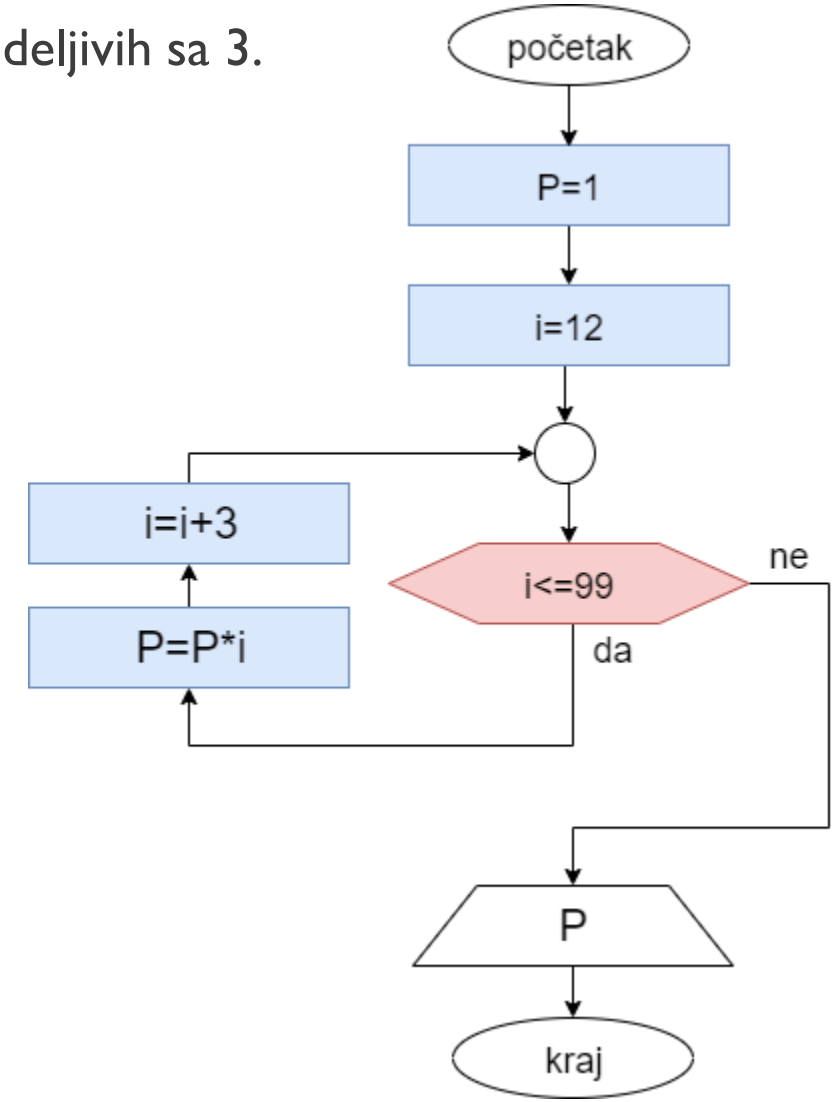
# PRIMER
# DO-WHILE CIKLIČNA STRUKTURA

- Sabrati trocifrene brojeve koji se završavaju sa 7.



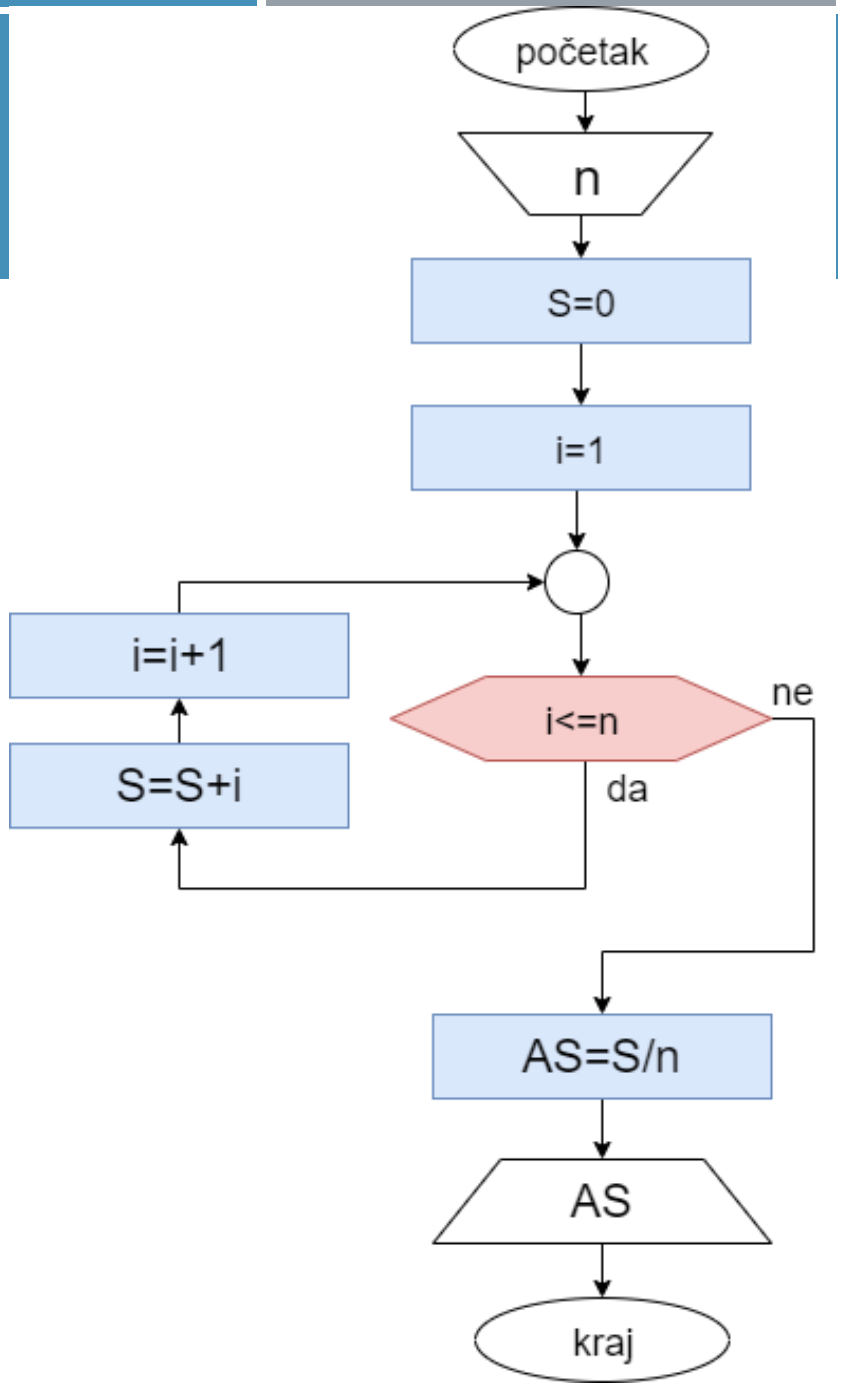
# PRIMER
# DO-WHILE CIKLIČNA STRUKTURA

- Odrediti prizvod dvocifrenih brojeva deljivih sa 3.



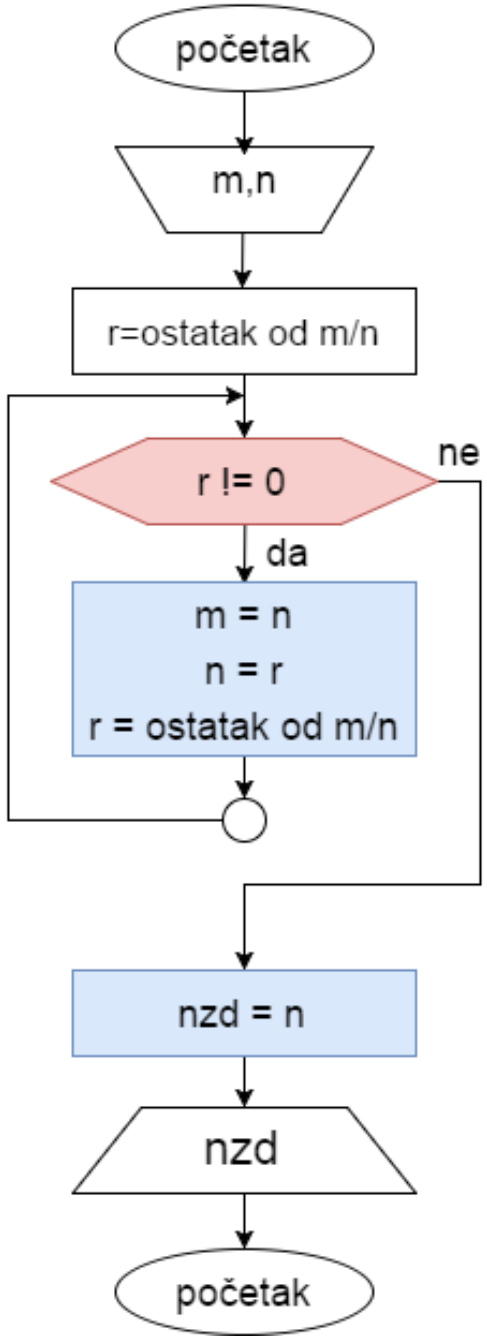
# PRIMER
# DO-WHILE CIKLIČNA

- Odfrediti aritmetičku sredinu prvih n prirodnih brojeva.



# 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;




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;

# PRIMER
# BROJAČKA CIKLIČNA STRUKTURA - FOR

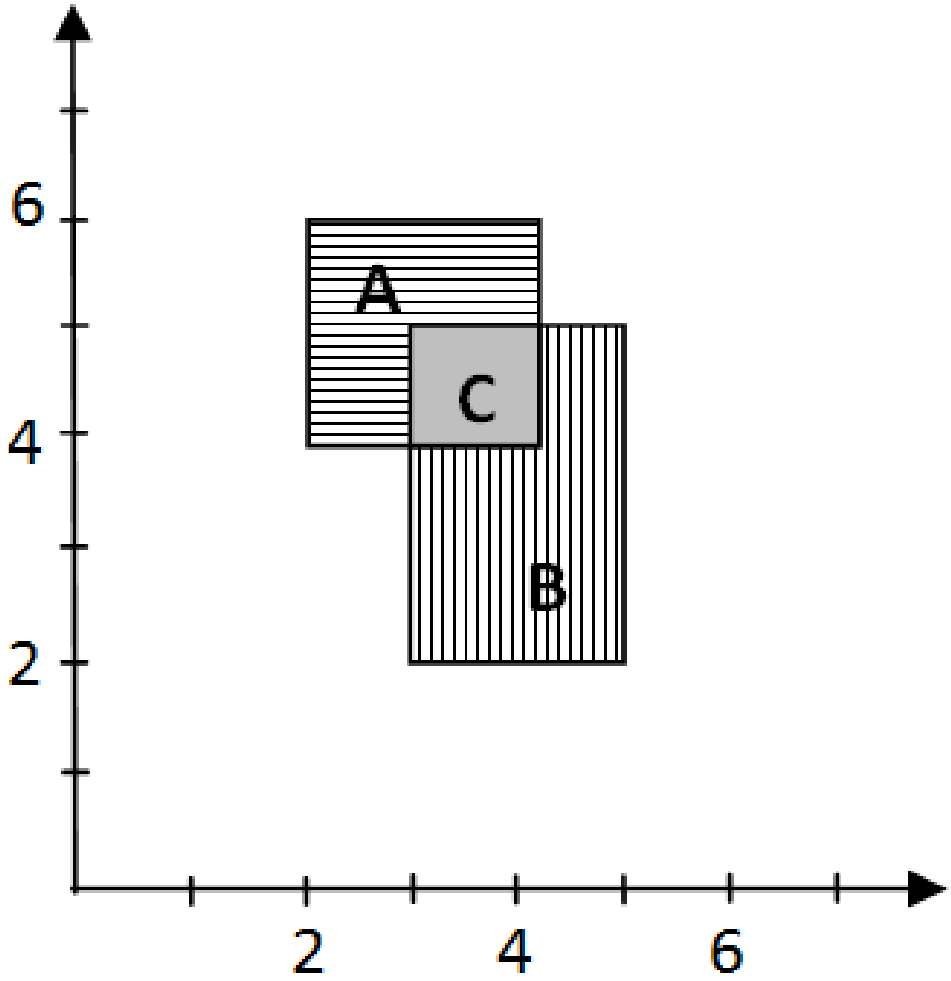
- ZADATAK 1. Rešiti prethodne DO-WHILE strukture pomoću FOR petlje.
- ZADATAK 2. Uneti n brojeva sa tastature. Odrediti koliko je unetih parnih brojeva, a koliko neparnih brojeva.
- ZADATAK 3. Odrediti proizvod prvih n neparnih prirodnih brojeva.
- ZADATAK 4. Odrediti sumu parnih brojeva i proizvod neparnih iz intervala od a do b.
- ZADATAK 5. Sastaviti program za izračunavanje sume:

$$S = \sum_{i=0}^{N} (X^{2i} + N)$$

# PRIMER
# DO-WHILE CIKLIČNA STRUKTURA

- ZADATAK 6. Nacrtati dijagram toka algoritma koji određuje sumu prvih 10 neparnih unetih brojeva. Brojeve unosi korisnik redom, sve dok ne unese 10 neparnih brojeva. Na izlazu štampati sumu brojeva i koliko je ukupno brojeva uneto (parnih i neparnih).

# GEOMETRIJSKI PROBLEMI
# PRIMERI



- ZADATAK 7. Nacrtati algoritam koji dozvoljava unos brojeva sve dok se ne unese 0 i štampati broj parnih unetih brojeva.
- ZADATAK 8. Unositi brojeve sve dok se ne unese 0 i štampati sumu i proizvod brojeva.
- ZADATAK 9. Unositi brojeve sve dok se ne unese 0, parne brojeve sabrati, neparne množiti. Na kraju štampati sumu parnih brojeva, proizvod neparnih brojeva i ukupan broj unetih brojeva.
- ZADATAK 10. Nacrtati algoritam za sumiranje brojeva. Sumiranje se vrši sve dok suma ne postane veća od unapred zadate vrednosti S.
- ZADATAK 11. Nacrtati strukturni dijagram toka algoritma koji dozvoljava unos brojeva sve dok broj unetih brojeva ne bude veći od 884 ili se unese broj koji je deljiv sa 10, i pri tome računa sumu unetih trocifrenih brojeva. Na izlazu štampati sve unete trocifrene brojeve i njihovu sumu.

# ALGORITMI I STRUKTURE PODATAKA

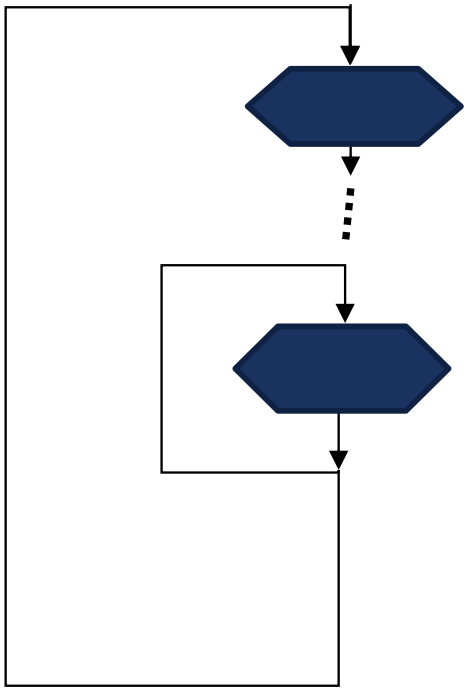
- PREDAVANJE 5-

# PRAVILA
# CIKLIČNA ALGORITAMSKA STRUKTURA

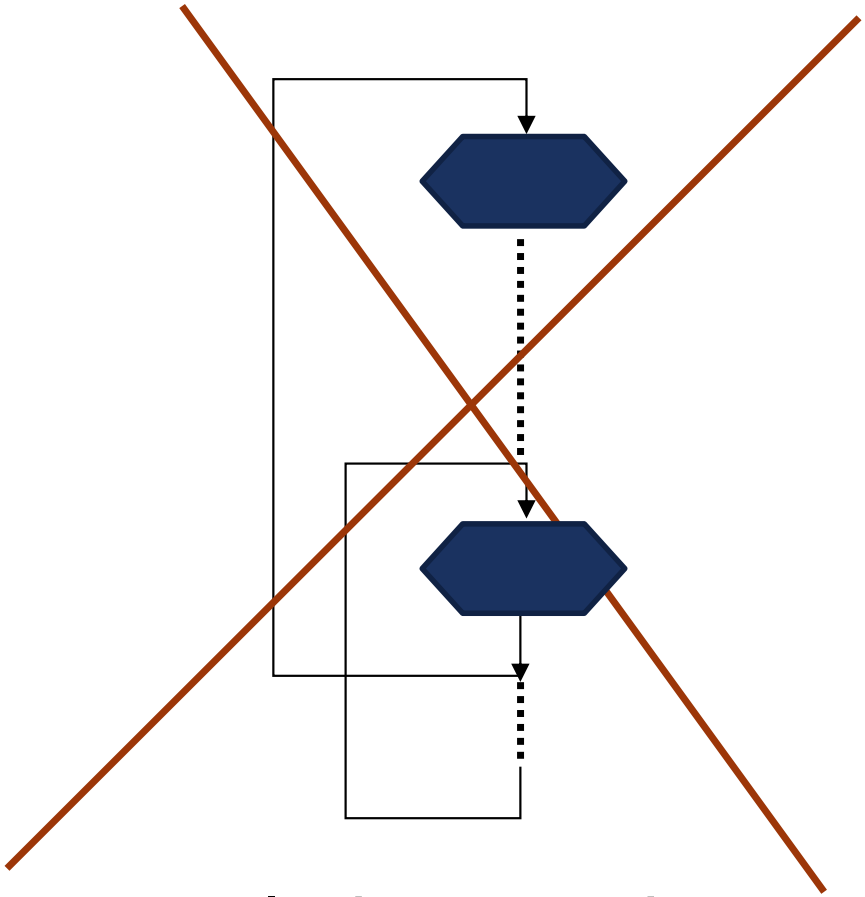
- Ciklusi mogu da obuhvataju i druge cikluse, tzv. **Ugnježdene petlje**.
- U tom slučaju važno je pravilo da spoljnja petlja mora da obuhvata sve instrukcije svake unutrašnje petlje.
- Ako petlja počne unutar *then* bloka ili *else* bloka, u tom bloku se mora i završiti!
- Dozvoljene su paralelne (ugnježdene) petlje.
- Nisu dozvoljene petlje koje se seku!

- PRIMER. Parne brojeve od 1 do n odštampati 3 puta, a neparne brojeve 2 puta.

# PRAVILA
# CIKLIČNA ALGORITAMSKA STRUKTURA



paralelne petlje



petlje koje se seku

# UGNJEŽDENE PETLJE

- ZADATAK 1. Sastaviti program za izračunavanje sume:

$$S = \sum_{i=0}^{M} \sum_{j=0}^{N} (10 * i + 2N * j)$$

- PRIMER 2: Kreirati algoritam koji ispisuje vreme u obliku **sati: minuti: sekunde**.
- PRIMER 3: Kreirati algoritam koji ispisuje brojeve od jedan do n u sledećem obliku.

**1**
**2 2**
**3 3 3**
**4 4 4 4**
**5 5 5 5 5**

# UGNJEŽDENE PETLJE

```
1
12
123
1234
12345
```

```
12345
1234
123
12
1
```

```
        1
      2 1 2
    3 2 1 2 3
  4 3 2 1 2 3 4
5 4 3 2 1 2 3 4 5
```

```
      6 5 4 3 2 1 2 3 4 5 6
    7 6 5 4 3 2 1 2 3 4 5 6 7
  8 7 6 5 4 3 2 1 2 3 4 5 6 7 8
```

# REKURZIJA

```
for (int i = 1; i <= rows; i++) {
        for (int j = 1; j <= (rows - i)*2; j++) //create initial spacing.
            System.out.print(" ");

        for (int k = i; k >= 1; k--) //creates left half.
            System.out.print(" "+ k);

        for (int k = 2; k <= i; k++)//creates right half.
            System.out.print(" "+ k);
```

```
                1
              2 1 2
            3 2 1 2 3
          4 3 2 1 2 3 4
        5 4 3 2 1 2 3 4 5
      6 5 4 3 2 1 2 3 4 5 6
    7 6 5 4 3 2 1 2 3 4 5 6 7
  8 7 6 5 4 3 2 1 2 3 4 5 6 7 8
```

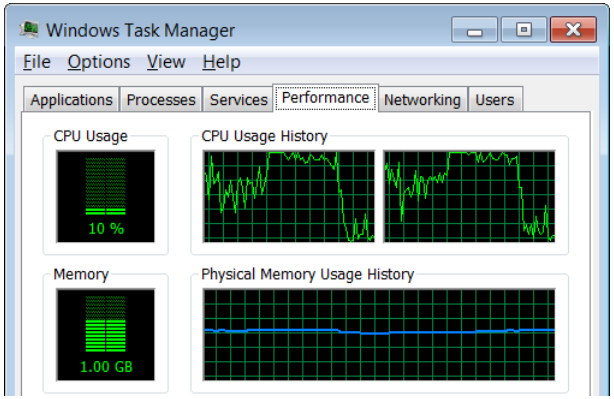
# ALGORITMI I STRUKTURE PODATAKA

- PREDAVANJE 6 -

# SLOŽENOST (KOMPLEKSNOST) ALGORITMA

- U teoriji složenosti (što nije isto što i teorija izračunljivosti) se izučava problematika složenosti, tj. kompleksnosti algoritma, u smislu zauzimanja resursa.
- Pod pojmom resursi se podrazumeva:
  - **PROSTOR** - količina zauzete memorije
  - **VREME** - količina potrošenog vremena procesora

- Složenost zavisi od veličine ulaznih podataka.
- Algoritmi se prave za rešenje opšteg problema, bez obzira na veličinu ulaza, ali sa druge strane razne ulazne veličine izazivaju da programi pisani na osnovu algoritma troše razne količine resursa.



# SLOŽENOST (KOMPLEKSNOST) ALGORITMA

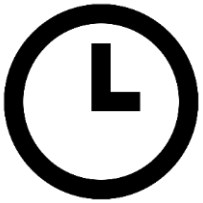
- **VREMENSKA SLOŽENOST** algoritma se iskazuje kao broj elementarnih koraka za obavljanje algoritma.
- Vremenska složenost naravno zavisi od veličine ulaza, a koja može biti izražena u bitovima ili nekim sličnim merilom.
- O notacija je jedan od metoda obeležavanja kompleksnosti algoritama.

# SLOŽENOST (KOMPLEKSNOST) ALGORITMA

- Algoritam (A) za uređivanje niza od $n$ elemenata je vremenske složenosti $n^2$. To znači da dvostruko veći broj elemenata zahteva četiri puta više vremena za uređivanje.
- Ako je, drugi algoritam (B) malo pažljivije napisan i brži je dvostruko, on će raditi dvostruko brže za bilo koju veličinu niza.
- Međutim, ako se programer namuči i osmisli suštinski drugačiji algoritam (C) za uređivanje, on stvarno može biti reda složenosti $n \cdot log(n)$.
- Algoritmi (A) i (B) su iste složenosti, jer se u notaciji sa velikim O obeležavaju sa $O(n^2)$, a u govoru se zovu 'algoritmi kvadratne složenosti', dok je algoritam (C) 'algoritam složenosti $n \cdot log(n)$'.
- Zaključak: algoritam (C) je najbolji sa stanovišta korišćenja vremena za velike setove ulaznih podataka.

# OCENA SLOŽENOSTI ALGORITMA

- Ocena (vremenske) složenosti algoritma sastoji se u brojaču računskih koraka koje treba izvršiti.
- Međutim, termin računska operacija može da podrazumeva različite operacije, na primer sabiranje i množenje, čije izvršavanje traje različito vreme.
- Različite operacije se mogu posebno brojati, ali je to obično komplikovano. Pored toga, vreme izvršavanja zavisi i od konkretnog računara, izabranog programskog jezika, odnosno prevodioca.
- Zato se obično u okviru algoritma izdvaja neki osnovni korak, onaj koji se najčešće ponavlja.



# SLOŽENOST (KOMPLEKSNOST) ALGORITMA

- Prostorna složenost se na isti način odnosi na funkciju zavisnosti zauzimanja memorijskog prostora u zavisnosti od veličine ulaza.
- Dešava se da je pronalaženje algoritma manje vremenske složenosti vezano sa povećanjem prostorne složenosti.
- Obično se prema složenosti algoritmi dele na algoritme:
  - Logaritamske složenosti
  - Linearne složenosti
  - Kvadratne složenosti
  - Polinomijalne složenosti
  - Eksponencijalne složenosti

# SLOŽENOST (KOMPLEKSNOST) ALGORITMA

- Kada imamo algoritam, potrebno je odlučiti na koji način ga implementirati.
  - Koje vreme izvođenja očekujemo?
  - Koliko memorije takav algoritam zahteva?
  - Kako će se ponašati algoritam u slučaju povećanja broja ulaznih podataka, tj. koliki će biti porast zahteva za vremenom i/ili memorijom?

- PRIMER
  - Razvili smo neki algoritam za obradu podataka o nekim entitetima i testirali funkcionalnost i brzinu sa 20 entiteta.
  - Odziv je bio vrlo brz i u zadanim granicama.
  - Da li će (i kako) taj algoritam raditi sa 10,000 entiteta, a kako sa 100,000, a da li sa 5,000,000 ?

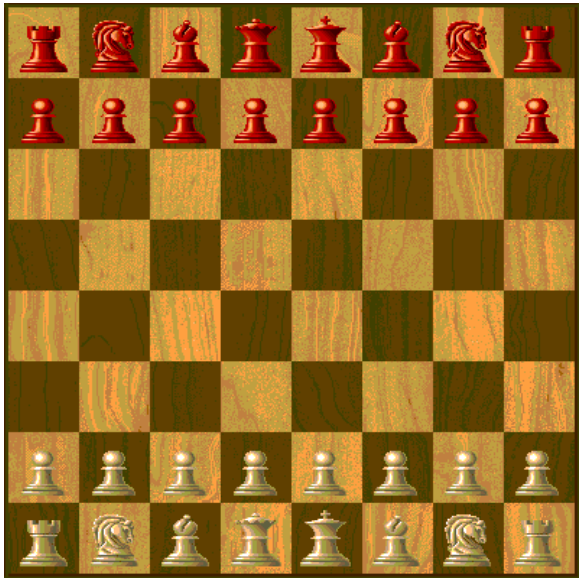
# SLOŽENOST (KOMPLEKSNOST) ALGORITMA

- Algoritam je upotrebljiv ako se rezultat dobije u konačnom vremenskom intervalu. Vreme izvođenja mora biti „razumno“.
- PRIMER: Algoritam koji bi birao prvi potez igrača šaha tako da ispita sve moguće posledice poteza, zahtevao bi milijarde godina na najbržem zamislivom računaru. Zašto?

20 mogućih prvih poteza belog
20 mogućih prvih poteza crnog
> 20 mogućih drugih poteza belog
> 20 mogućih drugih poteza crnog itd...

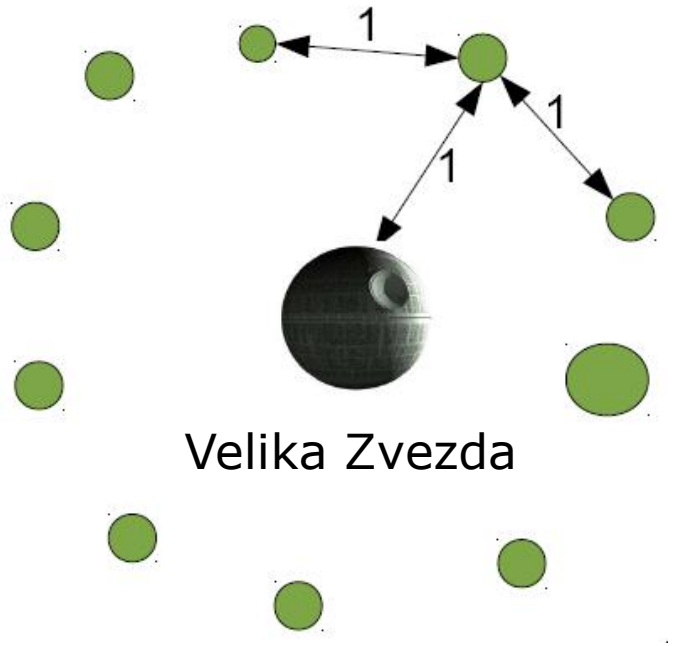
Za 10 poteza svakog igrača
barem 2020 kombinacija ~ 1026 poteza

Kad bi se jedna kombinacija analizirala 1μs,
to je 3.170.979.198.376 godina!



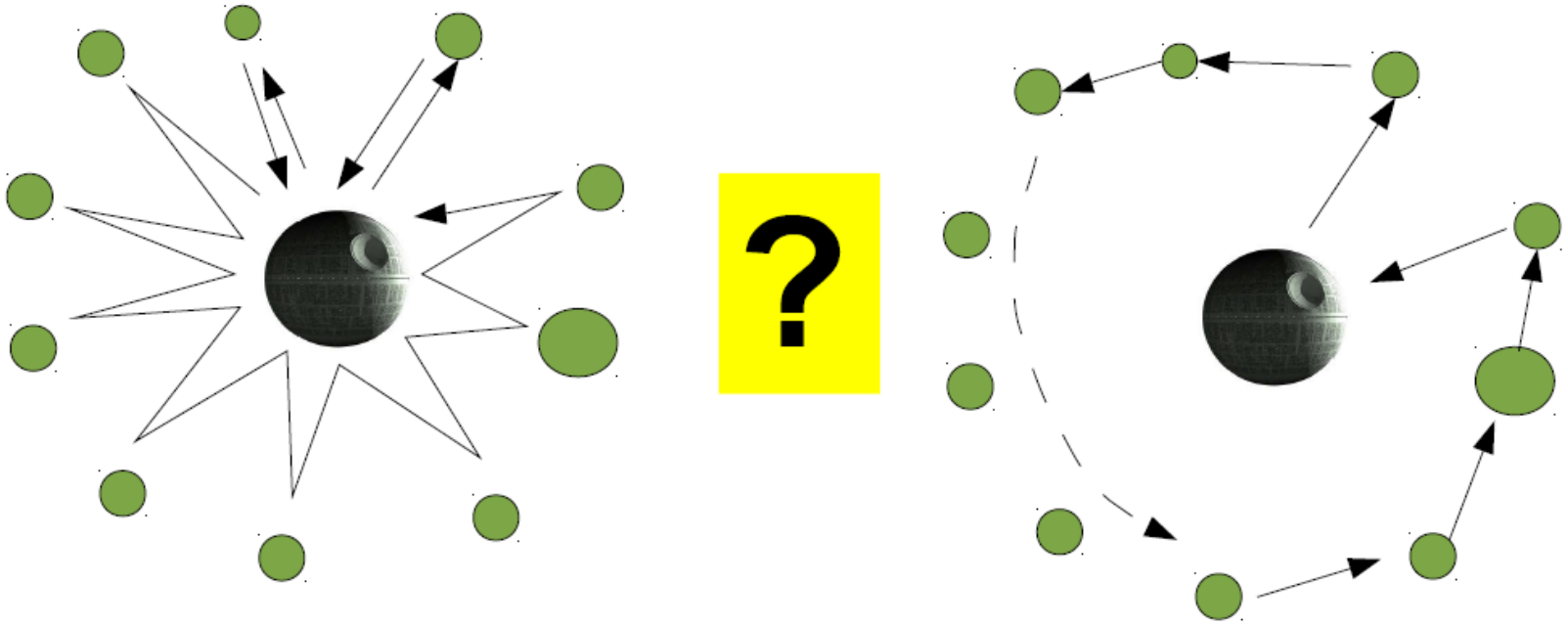
# SLOŽENOST (KOMPLEKSNOST) ALGORITMA

- Lokacije se nalaze u svim pravcima oko Velike Zvezde na prosečnoj udaljenosti od 1SG.
- Rastojanje između dve susedne lokacije je takođe u proseku 1SG, slično kao na slici (ali su u 3D prostoru).
- Potrebno je preneti po 1 tovar na sve lokacije uz uslov da se naprave ukupni najmanji troškovi.

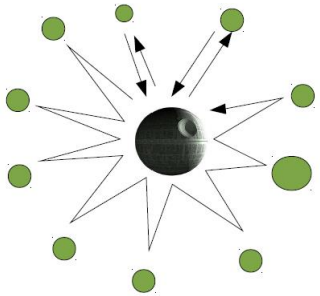


# SLOŽENOST (KOMPLEKSNOST) ALGORITMA

- Koju letilicu izabrati za isporuku navedenog tovara ?
- Šta se dešava kad imamo n lokacija, gde je n veliki broj, na primer 100?
  - Letelica A – nosivosti 1 tovara morala bi ići do svake lokacije i nazad.
  - Letelica B – nosivosti 125 tovara može utovariti svih 100 kontejnera i isporučiti na sve lokacije bez vraćanja.



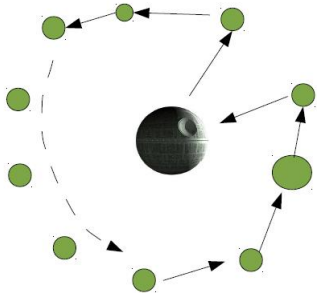
# PRORAČUN PREĐENOG PUTA



- Letelica A:

f1 = (1+1)+(1+1)+ ... +(1+1)=100 * 2 = 200
// 100-puta (ili n puta)

U slučaju n lokacija bi imali: f1(n)=2n



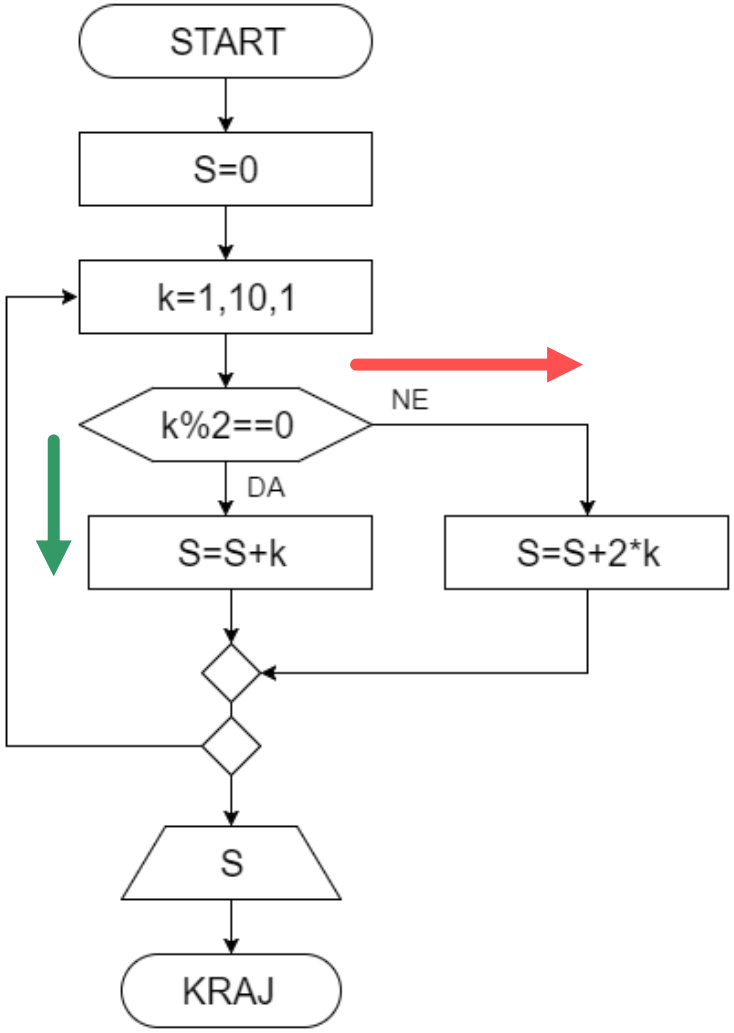
- Letelica B:

f2 = 1+1+ ... +1=100 // 100-puta (n puta)

U slučaju n lokacija bismo imali: f2(n)=n

- U primeru možemo proceniti da je zavisnost pređenog puta, pa samim tim i troškova, linearna u oba slučaja.
- Da bismo tačno procenili koji metod dostave (algoritam) je jeftiniji (ma šta to značilo), moramo ući u detalje svakog od njih (relativna potrošnja letelica, troškovi utovara i sl.).
- Ušteda može biti neverovatna.

# ANALIZA ALGORITMA
# PRIMER I



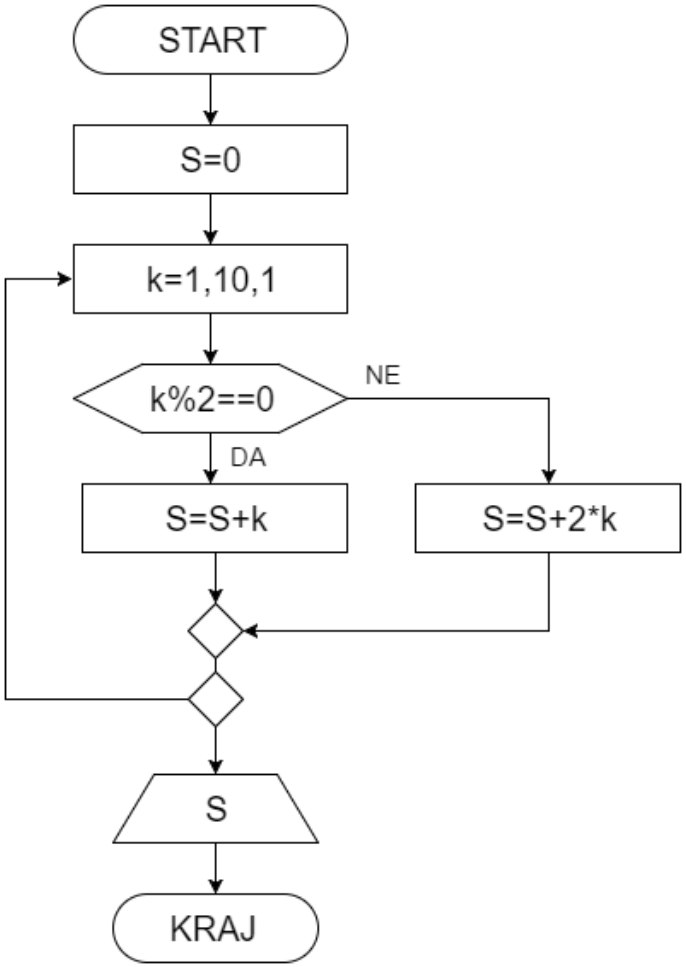

- Odrediti šta se dobija kao izlaz algoritma prikazanog na slici.
- ANALIZA:

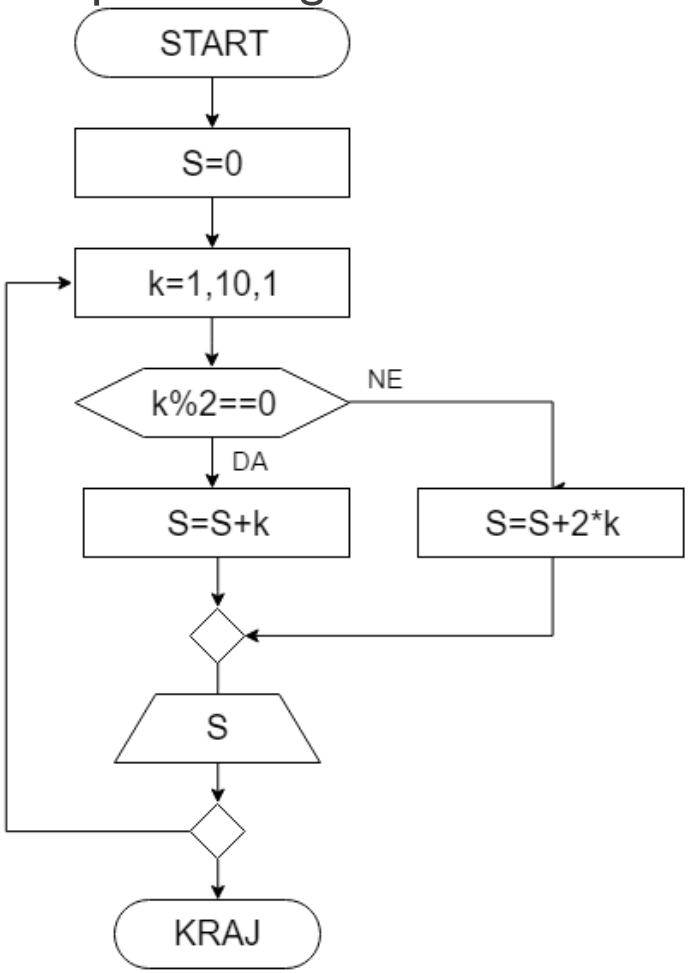
| | | |
|---|---|---|
| k = 1 | S=0+2*1 | S=2 |
| k = 2 | S=2+2 | S=4 |
| k = 3 | S=4+2*3 | S=10 |
| k = 4 | S=10+4 | S=14 |
| k = 5 | S=14+2*5 | S=24 |
| k = 6 | S=24+6 | S=30 |
| k = 7 | S=30+2*7 | S=44 |
| k = 8 | S=44+8 | S=52 |
| k = 9 | S=52+2*9 | S=70 |
| k = 10 | S=70+10 | S=80 |

- IZLAZ JE: 80

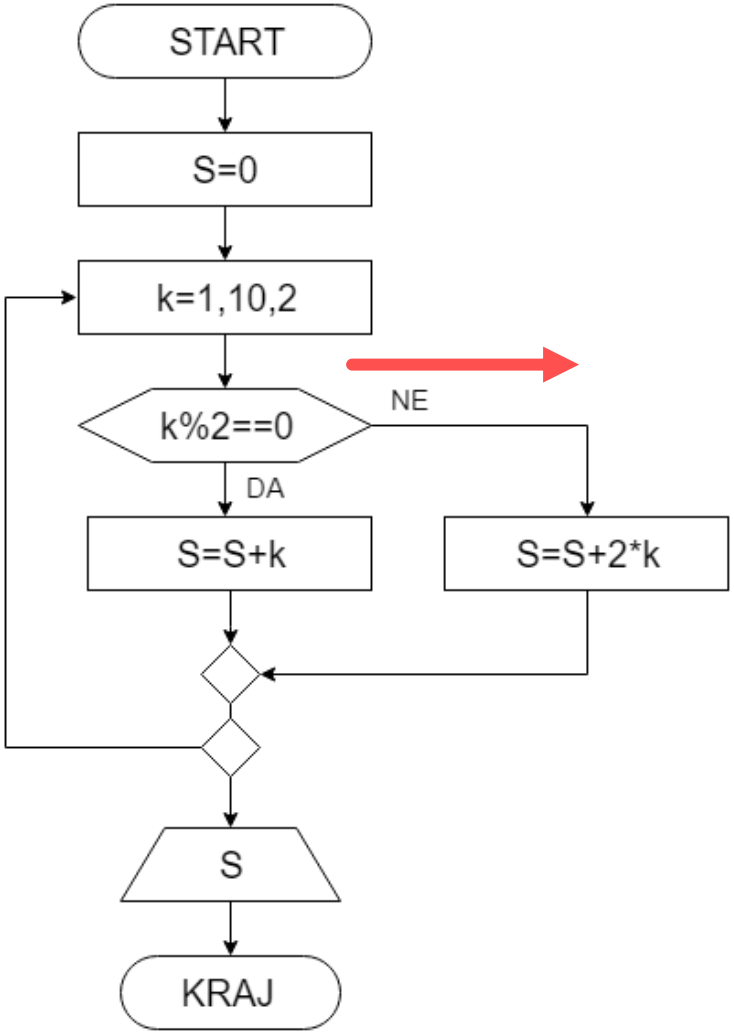
# ANALIZA ALGORITMA
# PRIMER 2

- Odrediti šta se dobija kao izlaz algoritma prikazanog na slici 1 i 2.



# ANALIZA ALGORITMA
# PRIMER 3

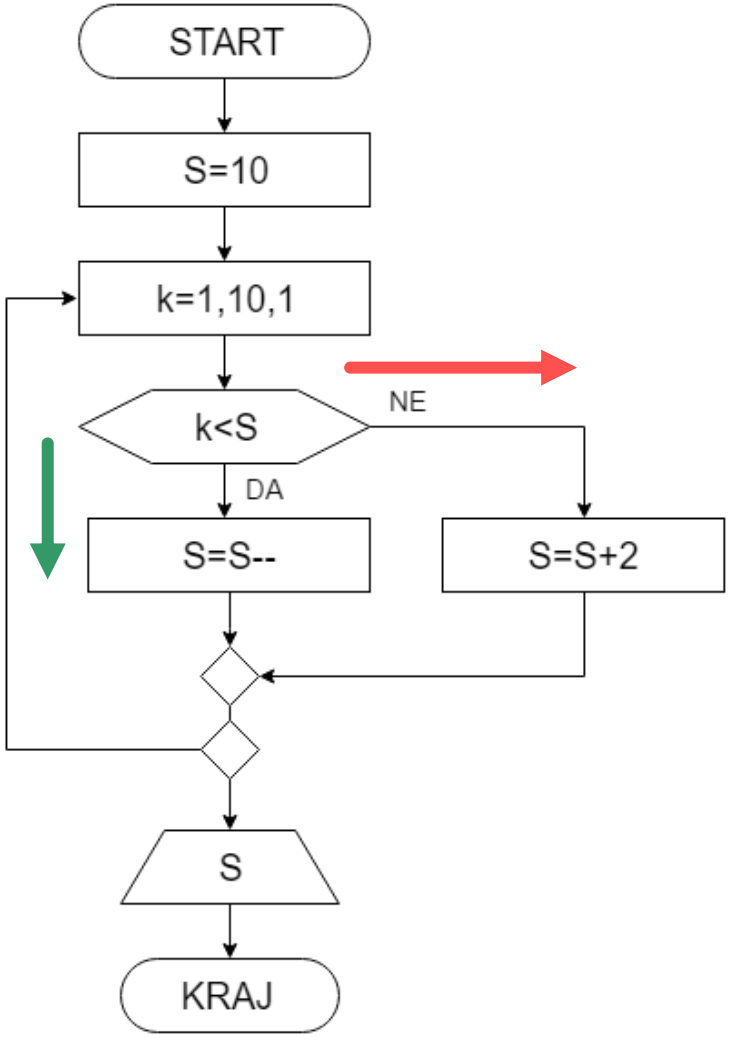



- Odrediti šta se dobija kao izlaz algoritma prikazanog na slici.
- ANALIZA:

| | | |
|---|---|---|
| k = 1 | S=0+2*1 | S=2 |
| k = 3 | S=2+2*3 | S=8 |
| k = 5 | S=8+2*5 | S=18 |
| k = 7 | S=18+2*7 | S=32 |
| k = 9 | S=32+2*9 | S=50 |

- IZLAZ JE: 50

# ANALIZA ALGORITMA
# PRIMER 4




- Odrediti šta se dobija kao izlaz algoritma prikazanog na slici.
- ANALIZA:

| | | |
|---|---|---|
| k = 1 | S=10-1 | S=9 |
| k = 2 | S=9-1 | S=8 |
| k = 3 | S=8-1 | S=7 |
| k = 4 | S=7-1 | S=6 |
| k = 5 | S=6-1 | S=5 |
| k = 6 | S=5+2 | S=7 |
| k = 7 | S=7+2 | S=9 |
| k = 8 | S=9-1 | S=8 |
| k = 9 | S=8+2 | S=10 |
| k = 10 | S=10+2 | S=12 |

- IZLAZ JE: 12

# ALGORITAM ZA USPEH

```
while(noSuccess)
{
        tryAgain();
        if(Dead)
                break;
}
```