ALGORITMI

- **Pojam algoritma**
- **Blok dijagram**

UVOD U ALGORITME

Sadržaj

- Pojam algoritma
- Primjeri algoritama
- Osnovna svojstva algoritama

Pojam algoritma

Što je algoritam? Grubo rečeno:

- **Algoritam = metoda, postupak, pravilo za rješenje nekog problema ili dostizanje nekog cilja.**

Ovo nije precizna definicija u matematičkom smislu, već samo opis preko drugih, sličnih pojmova, pri čemu je postupak najbliži.

Pojam algoritma (2)

Postupak asocira na konačan niz koraka koje treba napraviti za rješenje nekog problema

- Metoda se kao izraz često koristi u matematici, ali obično uključuje i tzv. beskonačne "postupke" koji tek na limesu daju rješenje (mat. analiza, numer.mat.)

Pojam algoritma (3)

Osnovna zadaća – razvoj efikasnih i točnih Algoritama

Intuitivno je jasno da efikasno znači brzo, a točno da je rješenje blizu "pravom" rješenju.

Primjeri algoritama

Npr. upute za uporabu (korištenje, rukovanje, instalaciju,...) tehničkih pomagala.

Upute za korištenje kartice na bankomatu

- umetnite karticu u čitač tako da magnetska traka bude s donje desne strane;
- u slučaju da to od Vas uređaj zatraži odaberite jezik;
- odaberite iznos ili uslugu;
- odgovorite želite li potvrdu;
- uzmite karticu;
- uzmite novac;
- uzmite potvrdu ako ste potvrdili da ju želite.

Primjeri algoritama (2)

Rješavanje linearne jednadžbe ax=b

- ako su i a i b jednaki nuli jednadžba ima beskonačno rješenja;
- ako je a jednak nuli i b različit od nule jednadžba nema rješenje;
- ako su i a i b različiti od nule jednadžba ima jedinstveno rješenje x=b/a.

Pojam algoritma

- Algoritam - postupak ili niz postupaka koje treba obaviti pri rješavanju određenog problema.

Načelo ekvifinaliteta - za rješavanje nekog problema ne mora postojati jedinstven algoritam.

Pojam algoritma

- Algoritam mora udovoljiti nekolicini kriterija:
  - Općenitost.
  - Konkretnost.
  - Svrhovitost.
  - Konačnost.
  - Efikasnost.
  - Ponovljivost.
  - Razumljivost.
  - Formaliziranost.
  - Instruktivnost.

Pojam algoritma

- **Općenitost**. Algoritam mora biti pogodan za rješavanje određenog tipa problema, a ne samo jednog konkretnog problema.
- **Konkretnost**. Algoritam mora prihvatiti konačan broj ulaznih veličina koje potpuno određuju konkretni problem koji treba riješiti.
- **Svrhovitost**. Algoritam mora dati barem jednu izlaznu veličinu, odnosno rezultat rješenja problema.
- **Konačnost**. Algoritam mora dati rješenje postavljenog problema u konačnom broju koraka odnosno postupaka.

Pojam algoritma

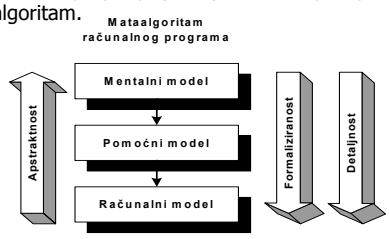
- **Efikasnost**. Postupak mora završiti u prihvatljivom vremenu i prihvatljivom utrošku drugih resursa.
- **Ponovljivost**. Ponovljeni postupak uz iste ulazne veličine mora dati isti rezultat, odnosno izlazne veličine.
- **Razumljivost**. Postupci određeni algoritmom moraju biti poznati izvršitelju.
- **Formaliziranost**. Svaki postupak mora biti jednoznačno i nedvosmisleno definiran.
- **Instruktivnost**. Postupci trebaju biti iskazani u formi naredbi izvršitelju.

Pojam algoritma

- **Algoritam** je uređeni skup jednoznačnih (nedvosmislenih), izvedivih koraka

Pojam algoritma

- Pojam algoritma danas se gotovo isključivo veže uz softver.
- Algoritam koji opisuje postupak stvaranja algoritma - metaalgoritam.



Pojam algoritma

- **Mentalni model** - visoki stupanj apstrakcije, mala detaljnost i formaliziranost.
  - Osnovne ideje kako riješiti problem, koje postupke koristiti i koji je približni redoslijed njihova izvođenja.
- **Pomoćni model** - posebnim tehnikama iskazan algoritam na dovoljnom stupnju formaliziranosti i detaljnosti kako bi se mogla provjeriti njegova logička ispravnost i olakšala izrada računalnog modela.

Pojam algoritma

- **Računalni model** - računalni program, niz logički povezanih instrukcija koje će omogućiti da se njihovim izvođenjem na računalu riješi konkretan problem.
- Nepreciznosti pri korištenju pojmova algoritam i program u praksi.

Načini predstavljanja algoritama

- Tekstualni
- Grafički (pomoću dijagrama toka)
- Pseudo kodom
- Strukturogramom

Tekstualni način

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

Grafičko predstavljanje algoritma

- Koriste se određeni grafički simboli za predstavljanje pojedinih aktivnosti u algoritmu
  - Ideja je posuđena iz teorije grafova
  - Algoritam se predstavlja usmjerenim grafom
    - čvorovi grafa predstavljaju aktivnosti koje se obavljaju u algoritmu
    - potezi ukazuju na slijedeću aktivnost koja se treba obaviti

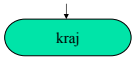
Blok dijagram

- **Blok dijagram** - grafički način predstavljanja algoritma skupom simbola koji označuju pojedine operacije, a njihov raspored i povezanost određuju slijed postupaka.

Grafičko predstavljanje algoritma

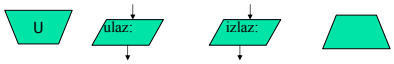
- Polazni čvor u usmjerenom grafu koji predstavlja algoritam nema dolaznih grana, a ima samo jednu izlaznu granu

- Krajnji čvor u grafu koji predstavlja algoritam koji nema izlaznih grana, a ima samo jednu dolaznu granu



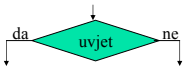
Grafičko predstavljanje algoritma

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



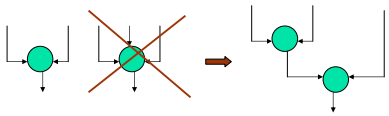
- Blok obrade je pravokutnog oblika

- Blok odkuke



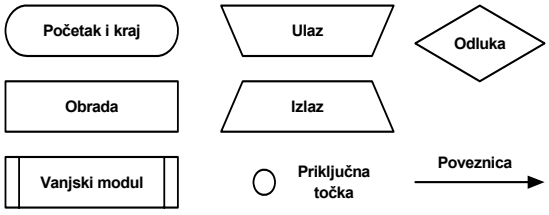
Grafičko predstavljanje algoritma

- Blok spajanja grana

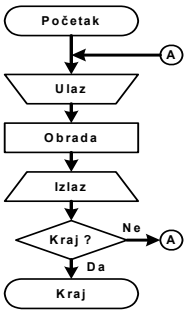


Blok dijagram

- Skup grafičkih simbola koji se koriste pri izradi blok dijagrama malen:



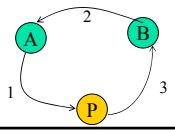
Blok dijagram



Opći blok dijagrama računalnog programa

Primjer1

- Zamijeniti sadržaje dvije memorijske lokacije, A i B
  - Rješenje – 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

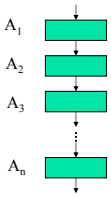


Osnovne algoritamske strukture

- Kombiniranjem blokova dobivaju se osnovne algoritamske strukture
  - Linijska (sekvenca,slijed)
  - Razgranata (selekcija, grananje)
  - Ciklička (iteracija, ponavljanje, petlja)
- Pomoću osnovnih algoritamskih struktura može se predstaviti svaki algoritam

Sekvenca,slijed

- linijska struktura koja se dobiva kaskadnim povezivanjem blokova obrade



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

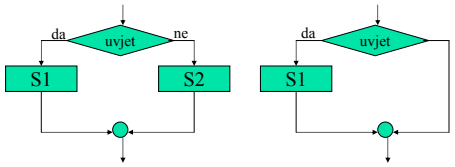
  varijabla:=*vrijednost*

  a:=b

  n:=n+1

Grananje (selekcija)

- Omogućuje uvjetno izvršenje niza algoritamskih koraka

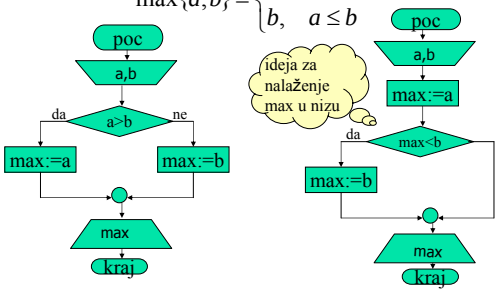


- Blokovi označeni sa S1 i S2 mogu sadržavati bilo koju kombinaciju osnovnih algoritamskih struktura.

Primjer1

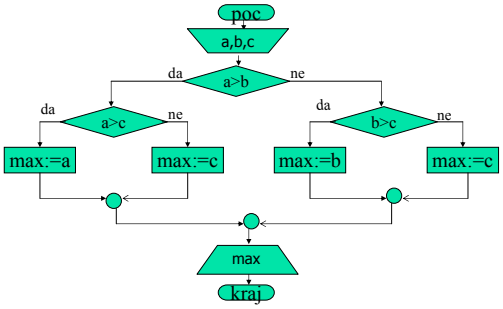
- Nacrtati dijagram toka algoritama kojim se određuje veći od dva zadata broja korištenjem formule

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

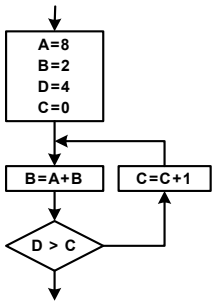


Naći maksimum od tri zadana broja a, b i c

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



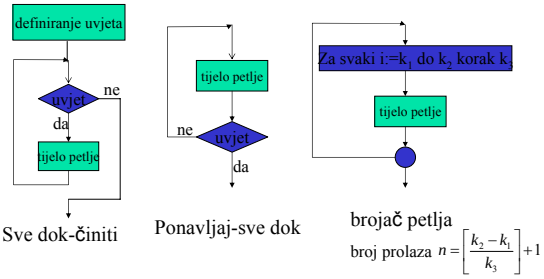
Blok dijagram



Primjer algoritma programske petlje

Petlja (ciklus)

- Omogćuje da se algoritamski koraci ponavljaju više puta.



Sve dok-činiti

Ponavljaj-sve dok

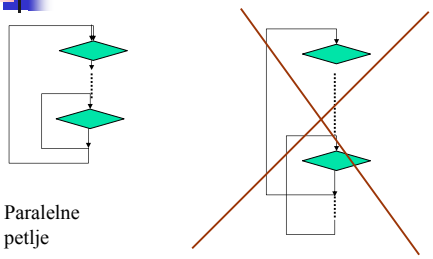
brojač petlja

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

Pravila

- Ako petlja počinjene unutar *tada* bloka ili *inače* bloka, u tom bloku se mora i završiti!
- Dozvoljene su paralelne (ugnježdene) petlje.
- Nisu dozvoljene petlje koje se sijeku!

Pravila



Paralelne petlje

petlje koje se sijeku

Predstavljanje algoritma pomoću pseudo koda

- Koristi se tekstualni način dopunjen formalizmom
  - svaka od osnovnih algoritamskih struktura se predstavlja na točno definiran način:
    - **Slijed ili sekvenca** se predstavlja kao niz naredbi dodjeljivanja odvojenih simbolom **;**

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

Predstavljanje algoritma pomoću pseudo koda (nast.)

- Grananje

```
Ako je (uvjet) tada
  niz_naredbi
inače
  niz_naredbi
Kraj Ako je;
```

ili

```
Ako je (uvjet)
  niz_naredbi
Kraj Ako je;
```

```
Ako je (a>b) tada
  max:=a
inače
  max:=b
Kraj Ako je;
```

ili

```
max:=a;
Ako je (max<a) tada
      max:=b
Kraj Ako je;
```

Grananje, primjer2

max(a,b,c)

```
ako je (a>b) tada
        ako je (a>c) tada
            max:=a
        inače
            max:=c
        kraj ako je;
    inače
        ako je (b>c) tada
            max:=b
        inače
            max:=c
        kraj ako je ;
    kraj ako je;
```

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

Sve dok (uvjet) činiti

    niz_ naredbi

Kraj sve dok;

Primjer:

```
r:="ostatak od m/n"
Sve dok (r≠0) činiti
    m:=n;
    n:=r;
    r:="ostatak od m/n";
Kraj Sve dok;

nzd:=n;
```

repeat

    niz_naredbi

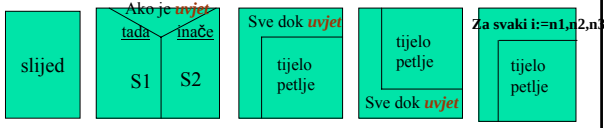
until (uvjet);

Primjer:

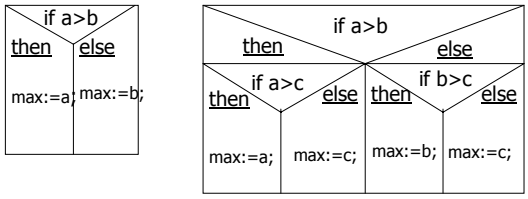
```
Ponavljaj
    r:="ostatak od m/n";
    m=n;
    n:=r;
Sve dok (r=0);
nzd:=m;
```

Strukturogrami

- Kombinacija grafičkog i pseudo koda;
  - Koriste se kao prikladna dokumentacija za već završene programe.
  - Program se piše uz popunjavanje određene geometrijske slike



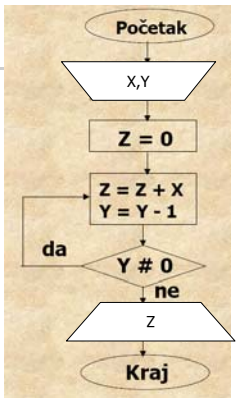
Strukturogrami - primer



Primjer:

Sastaviti algoritam za množenje dvaju proizvoljno zadanih prirodnih brojeva koristeći operaciju zbrajanja.

Rješenje: Neka su ulazne veličine
prirodni brojevi x i y s vrijednostima
kako slijedi : x = 14 i y = 3.



Algoritmi u svakodnevnom životu

- Algoritme koristimo svakodnevno a da toga često nismo niti svjesni.
- Na primjer

Što je dobar algoritam?

- U svim okolnostima daje točan rezultat
- Rješava problem u najkraćem mogućem vremenu
- Razumljiv je ostalima

KORACI ALGORITMA

Prvi korak - razumijevanje problema.

Drugi korak - detaljna razrada svakog pojedinačnog koraka.

Treći korak – provjera algoritma na granične uvjete

Primjer

- primjera algoritma napisanog hrvatskim jezikom (ne u programskom kodu).
- Određuje da li je zadani broj **n** paran ili neparan:

1. POČETAK
2. Pročitaj / Učitaj vrijednost n.
3. Podijeli n sa 2 i zapamti ostatak u pom.
4. Ako je pom 0 idi na stavku 7.
5. Ispiši „n je neparan broj“.
6. Idi na stavku 8.
7. Ispiši „n je paran broj“.
8. KRAJ

Algoritamske strukture:

- Slijedna (linearne ili sekvencijalne)
  - Početak i kraj
  - Definiranje varijabli i konstanti
  - Ulaz
  - Izlaz
  - Aritmetičke i logičke operacije
- Struktura bezuvjetnog skoka
- Struktura grananja (sadrži logičke operacije)
  - kombinira se sa: Slijednom strukturom
    - Strukturom bezuvjetnog skoka
- Struktura iteracije (ponavljanja ili petlje)

Slijedna struktura (moj prvi algoritam)

Početak — Zbroji dva broja x i y

Definiranje varijabli i konstanti
1. X=0, Y=0, S=0

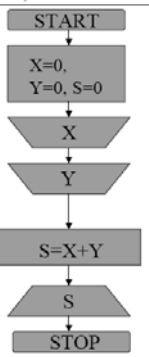
Ulaz
2. Učitaj x
3. Učitaj y

Aritmetička operacija
4. S=X+Y

Izlaz
5. Ispiši S

Kraj
6. Kraj

PSEUDO JEZIK



BLOK DIJAGRAM

Pozor! Kod pisanja algoritma u pseudo jeziku (kodu) linije se obično označavaju rednim brojevima

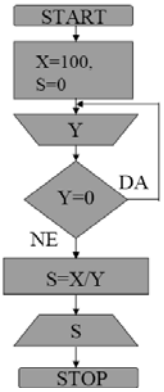
Struktura grananja – **uspostavljanje ponavljanja**)

u kombinaciji sa strukturom bezuvjetnog skoka

1. X=100, S=0
2. Učitaj Y
3. Ako je Y=0 onda idi na 2
4. S=X/Y
5. Ispiši S
6. Kraj

Podijeli X sa Y

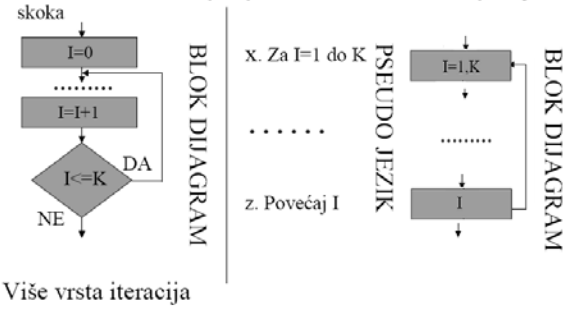
PSEUDO JEZIK



BLOK DIJAGRAM

Struktura iteracije (**uspostava ponavljanja**)

Evoluirala iz kombinacije slijedne strukture i strukture uvjetnog skoka



Više vrsta iteracija

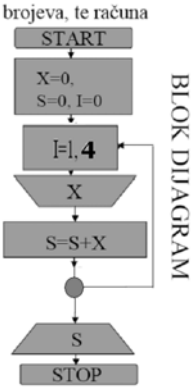
bavit ćemo se samo iteracijama s unaprijed definiranim konačnim brojem koraka

Struktura iteracije – **uspostavljanje ponavljanja**

Omogućava učitavanje konačnog broja brojeva, te računa kumulativ Unesi i zbroji četiri broja x

1. X=0, S=0, I=0
2. Za I=1 do 4
3. Učitaj X
4. S=S+X
5. Povećaj I
6. Ispiši S
7. Kraj

PSEUDO JEZIK

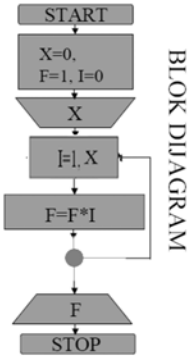


Struktura iteracije – **uspostavljanje ponavljanja**

Izračun Faktorijela (X!)

1. X=0, F=1, I=0
3. Učitaj X
2. Za I=1 do X
4. F=F*I
5. Povećaj I
6. Ispiši F
7. Kraj

PSEUDO JEZIK



Obračun telefonskih troškova

- ZADATAK
- Sastavite algoritam za obračunavanje telefonskih troškova na kraju mjeseca ako su poznati, broj potrošenih telefonskih impulsa, cijena jednog impulsa i iznos telefonske pretplate. U iznos telefonske pretplate uračunato je prvih 100 impulsa.

Obračun telefonskih troškova

POČETNO POZNATI PODACI:

- Broj potrošenih telefonskih impulsa (impulsi)
- Cijena jednog impulsa (cijena)
- Iznos telefonske pretplate (pretplata)
- Broj impulsa uračunat u pretplatu (100)

Obračun telefonskih troškova

ŽELJENI REZULTANTNI PODATAK:

- *iznos ukupnih telefonskih troškova.*

✷ Impulsi, cijena, pretplata, iznos su ***varijable***

- **VARIJABLA** - *veličina koja poprima vrijednosti iz skupa dopuštenih vrijednosti koje tijekom izvršavanja odredi algoritam .*

➢ 100 je ***konstanta*** - veličina stalne vrijednosti.

Obračun telefonskih troškova

ZAKONITOST ZA IZRAČUNAVANJE TROŠKA:

višak = impulsi - 100
trošak = pretplata + višak * cijena

Obračun telefonskih troškova

ALGORITAM

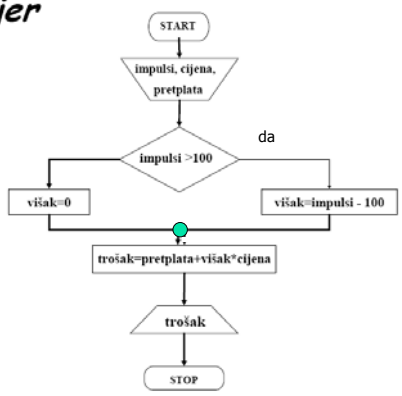
1. **UNESI PODATKE:** *impulsi, cijena, pretplata*
2. **AKO** *impulsi* > 100 **ONDA** *višak* = *impulsi* - 100 **INAČE** *višak* = 0
3. **trošak = pretplata + višak * cijena**
4. **ISPIŠI REZULTAT:** *trošak*

DIJAGRAM TOKA

Dijagram toka - preglednije pruža prikaz algoritma korištenjem standardnih grafičkih simbola

SIMBOLI	ZNAČENJE
	Početak / start, kraj / stop / zastoj
	Obrada, jedna ili više operacija, ako je rezultat operacija promjena vrijednosti
	U/I operacije
	Ulaz / izlaz podataka
	Mjesto izbora, grananja – prikazuje se operacija koja prikazuje istinitost nekog izraza
	Linija toka, pravac toka programa
	Povezivanje raznih dijelova u dijagramu toka

Primjer



Nekoliko tipičnih uzročnika logičkih grešaka

- *loš algoritam*
- *nepredviđene situacije*
- *podaci izvan domene*
- *nepotpuno poznavanje sintakse i semantike nekih naredbi*
- *preslikavanje značenja sličnih naredbi iz drugih jezika*
- *dvosmislenost u izrazima*
- *loš izbor imena varijabli…*

Algoritam se zapisuje u:

- Obliku pseudo (meta) jezika (govornog jezika koji oponaša programski jezik) i/ili
- Grafičkom obliku, tzv. dijagrama toka programa

Program se zapisuje u:

Programskom jeziku

Program se izvodi u:

Strojnom jeziku

Prevođenje