

# *Uvod u računarstvo*

## *2. predavanje*

Saša Singer

`singer@math.hr`

`web.math.hr/~singer`

PMF – Matematički odjel, Zagreb

# Sadržaj predavanja

- Uvod u algoritme:
  - Pojam algoritma.
  - Primjeri algoritama.
  - Osnovna svojstva algoritma.
  - Zapis i izvršavanje algoritama.
- Uvod u građu i funkcioniranje računala:
  - Što je računalo?
  - Instrukcije — operacije i podaci.
  - Osnovni dijelovi računala — ulaz, izlaz, memorija.
  - Izvršni dio računala — procesor.
  - Von Neumannov model računala.

# Uvod u algoritme

# Sadržaj

- Uvod u algoritme:
  - Pojam algoritma.
  - Primjeri algoritama.
  - Osnovna svojstva algoritma.
  - Zapis i izvršavanje 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.

- 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 (matematička analiza, numerička matematika).

# Pojam algoritma (nastavak)

Zašto je bitno znati što je algoritam?!

- Osnovna zadaća: razvoj **efikasnih** i **točnih** algoritama.
- Intuitivno je jasno da **efikasno** znači **brzo**, a **točno** da je rješenje **blizu** “pravom” rješenju. Detaljnije — poslije.
- Postoji i **precizna** matematička definicija pojma **algoritam**, ali ona nije sasvim jednostavna — potrebna je onima koji će se baviti **svojstvima** algoritama.
- Budući da je cilj kolegija naučiti **koristiti** algoritme za rješavanje raznih problema, dovoljno je (umjesto definicije) **opisati** osnovna **svojstva** algoritama.

# Primjer algoritma 1

Sva moderna tehnička pomagala imaju upute za **uporabu**, korištenje, rukovanje, instalaciju, ... ili kako vam drago.

**Primjer.** Upravo mi je dojadilo slušanje **zaštićenih** glazbenih CD-ova na računalu, jer:

- prvo se javi neki **odurni** “player”,
- a nakon toga počne **glazba** kôdirana u MP3 formatu **nevjerojatne** **nekvalitete** od 56 KB u sekundi.

**Prvo rješenje:** tehničko i nije za javno predavanje.

**Drugo rješenje:** kupio sam **priglupu** glazbenu CD liniju!  
Inače se može dogoditi sličan problem kao i na računalu — da stvar **ne radi**. Recimo, takvi su CD-players za automobile.

# Primjer algoritma 1 (nastavak)

I gdje je tu algoritam?

Što dolazi uz glazbenu liniju?

- **Upute** = kako **pospajati** sve razne dijelove, kablove i ostalo u funkcionalno radeću stvar!

Dakle, dotične **upute** su:

- **algoritam** za postizanje cilja = kako iz hrpe dijelova sastaviti **radeću** glazbenu liniju.

One “**druge**” **upute za uporabu** imaju smisla tek kad završimo s ovima!



# Primjer algoritma 1 (nastavak)

- To može izgledati ovako:
  - Kabel **A** (slijedi sličica) treba izvaditi iz vrećice,
  - utaknuti otraga u CD jedinicu u rupu **B** (vidi sličicu),
  - pažljivo stisnuti konektore,
  - zavinuti kabel,
  - prisloniti ga na stražnju površinu CD-jedinice
  - i na kraju ga (ipak) utaknuti u rupu **C** na pojačalu (nova sličica).
  - ...
- Sad se sličan postupak ponavlja jedno desetak puta!  
Za sat–dva, možda linija i proradi!

## Još primjera algoritama

Standardni primjeri algoritama:

- kulinarski recepti (pripravljanje jela),
- recepti za pripravu pića i koktela,
- uporaba bilo kakvih aparata (izbor operacija, ... ),
- rješavanje matematičkih zadataka!

# Opis algoritma

Općenito, kako moraju izgledati upute i iz čega se sastoje?

Opći izgled je nešto poput:

1. korak

2. korak

⋮

zadnji korak

Drugim riječima, algoritam se sastoji iz niza koraka i to, naravno konačnog broja, koje treba izvršiti da bi se postigao cilj, odnosno, riješio problem.

Svaki pojedini korak algoritma je instrukcija ili naredba (može i akcija) koju treba napraviti (izvršiti).

# Zapis i izvršavanje algoritama

- Uobičajeno, instrukcije se
  - pišu jedna ispod druge i
  - izvršavaju tim redom.
- Ipak, postoje i instrukcije koje mijenjaju standardni redoslijed izvršavanja. Primjeri:
  - ako se dogodi “to i to”, onda otiđi na korak “taj i taj”, ili
  - ponovi “neke korake” sve dok je ispunjen uvjet “taj i taj”, ili
  - ponovi “neke korake” određeni broj puta.

# Vrste instrukcija

Grubo govoreći, postoje **dvije** vrste instrukcija za kontrolu redoslijeda operacija (tj. izvršavanja ostalih instrukcija):

- **Uvjetne** = izbor jedne od mogućih alternativa.
- **Petlje** = ponavljanje nekog bloka naredbi pod kontrolom uvjeta ili brojača.

Ovakve instrukcije su **ključne** u programiranju.

Razlog?

# Izgled instrukcije

● Svaka instrukcija ima **sličan oblik** i sastoji se iz **2 dijela**:

● **što treba napraviti = operacija**

● **nad čim to treba izvršiti = objekt nad kojim se obavlja operacija**

● Primjer: izvadi kabel (sličica) iz vrećice.

● Isti oblik imaju i instrukcije na računalu:



● Primijetite da se **ista** instrukcija može obavljati nad **raznim** podacima.

## Opis algoritma (nastavak)

- Algoritam bi trebao raditi nad “općenitim” podacima, tj. bitno je da se samo radi o istom postupku.
- Na primjer, puno je bolje napisati algoritam koji nalazi rješenja “opće” kvadratne jednadžbe

$$ax^2 + bx + c = 0$$

za proizvoljne  $a, b, c \in \mathbb{R}$ , uz  $a \neq 0$ , nego onaj koji nalazi rješenja “konkretne” jednadžbe

$$x^2 - 3x + 2 = 0.$$

## Opis algoritma (nastavak)

- U slučaju već spomenute kvadratne jednadžbe, rješenja su dana formulom:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

- Primijetite da **izračunata** rješenja mogu biti **kompleksna**.
- Što su ovdje instrukcije?
- Napomene o točnosti (bolje rješenje — kasnije)!
- Rješivost jednadžbi višeg stupnja. Do stupnja 4 ide “formulama”, preko 5 **nema** općih formula.



# Svojstva algoritma

- Dakle, algoritam izvodi neke operacije nad nekim podacima u obliku niza koraka i daje neko rješenje.

- Gruba skica:



- Svojstva algoritma su:

- ima (ili nema) ulazne podatke,
- ima izlazne podatke,
- završava u konačnom vremenu,
- uvijek je nedvosmisleno definiran,
- mora biti efikasan (završiti u razumnom vremenu).

# Ulaz/izlaz

## ● Ulaz:

- Svaki algoritam ima 0 ili više, ali konačno mnogo ulaznih podataka.
- Njih moramo izabrati iz neke dozvoljene klase ulaznih objekata.
- Algoritmi s 0 ulaza nisu česti, ali ima ih. Opisuju fiksni postupak. Recimo: provjeri je li 327 prost broj ili riješi konkretnu kvadratnu jednadžbu.
- Ako algoritam ima više različitih objekata na ulazu, kažemo da je općenit, jer rješava cijelu klasu problema. Recimo: kvadratna jednadžba s općim  $a$ ,  $b$  i  $c$ .

# Ulaz/izlaz (nastavak)

- Izlaz:

- Svaki algoritam **mora** imati **bar jedan** izlaz, jer inače nije ostavio trag svog izvršavanja.
- Ova prva dva svojstva pričaju o objektima na ulazu i izlazu, ali ništa ne kažu o tome **kako** se iz jednog dolazi do drugog.
- Ostala svojstva nešto govore o “crnoj kutiji” na ranijoj skici



# Konačnost

## ● Konačnost:

- Svaki algoritam **mora** završiti u konačno mnogo koraka za **svaki** ulaz.
- U programu uvijek treba provjeriti je li ulaz **korektno** zadan. Na primjer, u kvadratnoj jednadžbi koeficijent  $a$  može biti  $0$ . U tom slučaju jednadžba nije kvadratna, a u formuli za rješenje pojavit će se dijeljenje s  $0$ !
- U praksi treba predvidjeti sva moguća korisna ograničenja i **ugraditi** ih u program. Na primjer, program koji očitava temperaturu  $T$  vode u posudi trebao bi imati ograničenje da je  $0 \leq T \leq 100$ .

# Definiranost i nedvosmislenost

- Kad projektiramo algoritam, **ne znamo** odmah na početku od kojih se koraka sastoji čitav postupak za rješenje problema.
- Obično sa problem rastavi na nekoliko većih cjelina, koje rješavamo pazeći na međuovisnost potproblema. Ako su cjeline velike, one se mogu rastavljati na manje dijelove ... Ovakva se metoda obično zove metoda postupnog profinjavanja (engl. stepwise refinement).
- Dokle treba postupno profinjavati? Ovisi o **izvršitelju** algoritma, tj. koje instrukcije on prepoznaje i koje može izvršavati.

# Definiranost i nedvosmislenost (nastavak)

- Definiranost i nedvosmislenost:
  - Algoritam se sastoji od niza osnovnih (elementarnih, primitivnih) instrukcija i mora biti jednoznačno i nedvosmisleno definiran za izvršitelja algoritma.
  - **Primjer:** Ja volim tropetinsku ( $\frac{3}{5}$  jabuka i  $\frac{2}{5}$  tijesta) pitu od jabuka. Ako je moja “bolja polovica” kod kuće, onda će moja molba: “Napravi mi pitu od jabuka” biti uslišena. Za nju je pravljenje pite od jabuka elementarna instrukcija, (jer je ona dovoljno moćan izvršitelj). Dakle, za moj algoritam kako se dočepati pite to je elementarna instrukcija.

## Definiranost i nedvosmislenost (nastavak)

- Naravno, postoji i lošije rješenje (po mene). Ako ona nije u blizini, a ja ipak želim pitu od jabuka, onda je instrukcija “ispeci pitu od jabuka” za mene **presložena**, pa moram zaviriti u kuharicu, gdje su dane jednostavne instrukcije (vidi prilog).
- Dakle, kad za rješavanje problema koristimo računalo, potrebno je znati što ono zna i može napraviti.
- Za zapisivanje algoritam postoje jezici ključno određeni snagom računala.

# Efikasnost

- Efikasnost:
  - Algoritam mora završiti u **razumnom** vremenu, što je bitno jači zahtjev od konačnosti.
  - Recimo, **500** godina **nije** razumno vrijeme!
  - Ima li takvih algoritama? **Ima!**
  - Tzv. **NP–potpuni** problemi, za koje ne postoje **efikasni** algoritmi. Primjer: trgovački putnik.
- Napomena: postoje problemi za koje **ne postoji** algoritam za njihovo rješenje — tzv. algoritamski nerješivi problemi. (Primjer.)



# Principi rada računala (Građa i funkcioniranje)

# Sadržaj

- Uvod u građu i funkcioniranje računala:
  - Što je računalo?
  - Instrukcije — operacije i podaci.
  - Osnovni dijelovi računala — ulaz, izlaz, memorija.
  - Izvršni dio računala — procesor.
  - Von Neumannov model računala.
  - Turingov stroj (sljedeći put).

# Gradnja i funkcioniranje računala

## ● Što je računalo?

● Računalo = stroj za izvršavanje algoritama.

## ● Prisjetimo se oblika instrukcija:



što = operacija      s čim = podatak ili operand.

● Operacije koje računalo izvršava su osnovne strojne instrukcije.

● Podaci s kojima “zna” raditi su osnovni ili elementarni podaci.

# Gradnja i funkcioniranje računala (nastavak)

- **Želja:** imati računalo koje dozvoljava složene instrukcije na složenim podacima.
- **Problem:** čisto tehnološki kako to brzo realizirati. Zbog toga postoje ograničenja koja diktiraju opći izgled današnjih računala.
- Zašto matematičar uopće mora nešto znati o građi računala — zato da bi ih mogao efikasno koristiti, pisati **brze** i **točne** algoritme (ograničenja proizlaze iz fizičke građe).

# Ulaz, izlaz, memorija

- Budući da algoritam ima ulaz i izlaz, mora to imati i računalo.
- **Ulazni dio:** čita podatke s nekog medija.
- **Izlazni dio:** mehanizam koji će na neki medij napisati podatke.
- Algoritam izvršava neke instrukcije nad podacima koje je učitao, tj. iz njih pravi nove podatke koje može više puta koristiti. Prema tome, računalo mora moći te podatke negdje spremiti — treba imati neku **memoriju** u koju može spremiti podatke i iz nje čitati.

## Izvršni dio računala

- Potrebno imati i **izvršni dio** koji izvršava instrukcije nad podacima.
- **Dilema**: trebe li mi **poseban stroj** za svaki algoritam posebno ili je bolje imati **stroj opće namjene**?
- **Poseban stroj** izvršava samo jedan algoritam koji je **tvrd** ugrađen u arhitekturu stroja. Nije besmisleno imati takav stroj. Ako malo šire gledamo to nas asocira na upravljačke sklopove u ulazno–izlaznim jedinicama poput CD čitača ili “pržilice”.
- **Računalo opće namjene** je složenije jer s algoritmom mora postupati slično kao s podacima. Svaki algoritam mora **učitati, spremiti, izvršiti**.

## Izvršni dio računala (nastavak)

- Svako računalo opće namjene stalno izvršava jedan meta-algoritam (jer se stalno vrti – ne završava izvođenje dok ne ugasimo stroj). To je ono što obično zovemo **operativni sustav**.
- **Pitanje:** gdje se spremaju algoritmi, odnosno programi?
- Mogli bismo imati posebnu memoriju za programe, ali nema potrebe za tim, pa se programi spremaju u **istu** memoriju kao i podaci.
- Spremanje podataka i algoritama u istu memoriju ključna je stvar **von Neumannovog modela** računala.

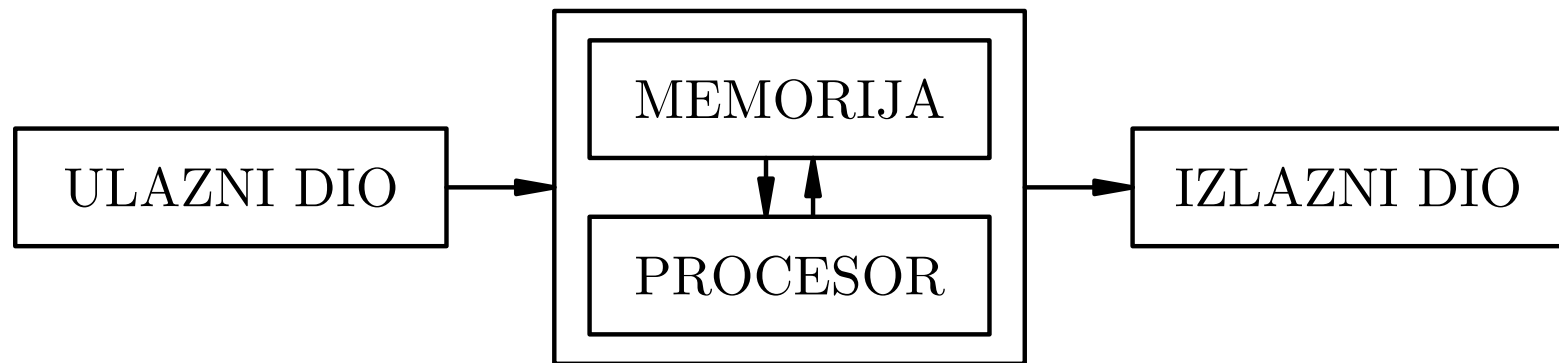
## *von Neumannov model*

- Ideja je nastala početkom četrdesetih godina prošlog stoljeća, a objavljena je tek nakon 2. svjetskog rata.
- Za vrijeme rata poslužila je kao podloga za gradnju prvih računala opće namjene, koja su računala balističke tablice (posebno za brodove) i jasno je da je bila strogo čuvana tajna.



## Izvršni dio računala (nastavak)

- Sasvim općenito, ako izvršni dio računala zovemo standardnim imenom – **procesor**, onda shematski model računala izgleda ovako:



- Skraćena slika bitnog dijela računala:

