PROGRAMIRANJE 1

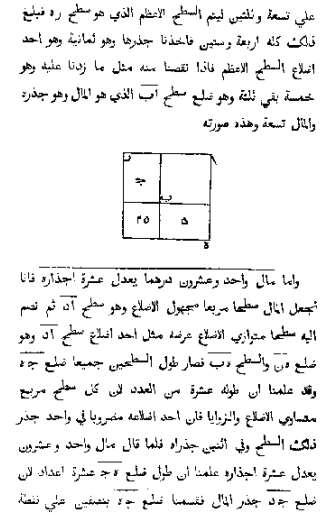
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
SVEUČILIŠTE U SPLITU

python™

Algoritam

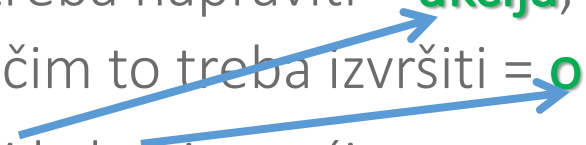
- **Algoritam** je:
  - precizno opisan način rješenja nekog problema
  - točno propisani postupak za izvršavanje, određenim redoslijedom, definiranih postupaka koji vode do rješenja postavljenog zadatka
  - skup pravila, definiranih s ciljem rješavanja određenog zadatka
- Svako pojedinačno pravilo, iz skupa pravila definiranih za rješavanje zadatka zove se **algoritamski korak**
  - Kroz te korake se vrši transformacija ulaznih vrijednosti parametara u izlazne.

- Priča o algoritmima
  - Bagdad - **Muhammed ibn Musa al Khowarizmi** pisac, matematičar, astronom i geograf
  - 825. godine napisao je knjigu **Kitab al jabar w'al-muqubala** u kojoj je opisao postupke za računanje u indijskom brojevnom sustavu.
  - prikazao rješenja nekih aritmetičkih problema u obliku uputstava koja su se sastojala od točno određenih pravila
  - ta uputstva (algoritmi) danas su postala važno i samostalno područje računalne znanosti
  - original na arapskom nije sačuvan, a latinski se prijevod proširio Europom pod naslovom **Algoritmi de numero Indorum** (“Al-Khwarizmi o indijskim brojevima").
  - prema tom naslovu postupke za sustavno rješavanje problema danas nazivamo **algoritmima**.



- Algoritam se sastoji od **konačnog** niza koraka koje treba izvršiti da bi se riješio problem.
  1. korak
  2. korak
  ...
  n. korak
- Svaki pojedini korak algoritma je akcija koju treba napraviti.
- Promjena poretka izvršavanja koraka algoritma najčešće dovodi do neželjenih rezultata
- Većina zadataka se može riješiti na više različitih načina pa je za njihovo rješenje moguće napisati više različitih algoritama.
- Autor algoritma nastoji pronaći algoritam koje najbrže, najučinkovitije i najsigurnije dovodi do rezultata.

- Svaki korak ima sličan oblik i sastoji se iz 2 dijela:
  - što treba napraviti = **akcija**,
  - nad čim to treba izvršiti = **objekt** nad kojim se obavlja akcija
- Npr. Izvadi kabel iz vrećice



- Ista akcija se može obavljati na(d) raznim objektima
  - Npr. Izvadi vijak iz vrećice

- Algoritam bi trebao raditi nad “općenitim” podacima.
- Konkretne podatke zadajemo svaki put kad izvršavamo algoritam i možemo ih mijenjati.

- Na primjer, puno je bolje napisati algoritam koji nalazi rješenja “opće” kvadratne jednadžbe $ax^2 + bx + c = 0$, nego onaj koji nalazi rješenja “konkretne” jednadžbe $x^2 - 3x + 2 = 0$.

- Algoritam izvodi akcije nad podacima u obliku niza koraka i daje rješenje.

- Ulaz:
  - Svaki algoritam ima 0 ili više, ali **konačno mnogo** ulaznih podataka
  - Ako algoritam ima više ulaznih podataka, kažemo da je općenit, jer rješava cijelu klasu problema (kvadratna jednadžba s općim a,b i c)
- Izlaz:
  - Svaki algoritam mora imati **najmanje jedan** izlaz, jer inače nije ostavio trag svog izvršavanja.
  - To je traženo "rješenje" našeg problema.

- Karakteristike algoritma:
    1. Izvediv
    2. Jednoznačan
    3. Korektan
    4. Uporabljiv
    5. Učinkovit

- Koraci moraju biti **izvedivi** i **jednoznačni**
  - Algoritam se sastoji od niza osnovnih akcija, koje moraju biti jednoznačno i nedvosmisleno definirane – za onoga tko izvršava algoritam.
  - Primjeri za nedopuštene akcije:
    - izračunaj 5/0
    - uvećaj *x* za 6 ili 7

- Algoritam je **korektan** ako za sve dozvoljene ulaze omogućuje određivanje rezultata i ako je taj rezultat ispravan.

- Algoritam je **uporabljiv** ako se dobije rezultat u **konačnom** vremenu
  - **Vremenska složenost ili kompleksnost algoritma** je trajanje algoritamskog procesa, izraženo kao broj osnovnih operacija koje treba obaviti.
  - Svaki algoritam mora završiti u konačno mnogo koraka, za svaki dozvoljeni ulaz.
  - Uvijek treba provjeriti je li ulaz korektno zadan.
  - U praksi treba predvidjeti sva moguća ograničenja ulaza.

- Mora biti **učinkovit**, tj. završiti u **razumnom** vremenu
  - Algoritam mora završiti u razumnom vremenu, što je bitno jači zahjev od konačnosti.
  - Npr. 500 godina nije razumno vrijeme, ali je konačno!
    - Zbrajanje cijelih brojeva je učinkovito
    - Dijeljenje realnih brojeva nije jer se može pojaviti broj s beskonačno mnogo znamenki, npr. 10/3 = 3.3333333...
      - Algoritam postaje učinkovit tek ako se broj znamenki unaprijed ograniči

- Jedan od najpoznatijih problema za čije rješavanje, u nekim situacijama, nema učinkovitog algoritma je **problem trgovačkog putnika** - pronaći najkraći mogući put kojim bi trgovački putnik obišao dani skup gradova.
- Pronalaze se svi mogući putovi, računaju njihove duljine i uspoređuju se, te se pronalazi najkraći put.
  - Za deset gradova, broj svih mogućih puteva je 10! (1*2*3*4*5*6*7*8*9*10 = 3 628 800 puteva).
  - Za dvadeset gradova, broj svih mogućih puteva je 20! (2 432 902 008 176 640 000).
- Ako pretpostavimo da računalo može izračunati duljine milijun puteva u sekundi, potrebno je preko 77 000 godina za pronalazak najkraćeg puta za 20 gradova.

- **Metoda postupnog profinjavanja** (eng. stepwise refinement)
  - Kad definiramo algoritam, ne znamo odmah od kojih se koraka sastoji čitav postupak za rješenje problema.
  - Obično se problem rastavi na nekoliko većih cjelina, koje rješavamo pazeći na meduovisnost potproblema.
  - Ako su te cjeline prevelike za izravno rješavanje, one se mogu rastavljati na još manje dijelove, …

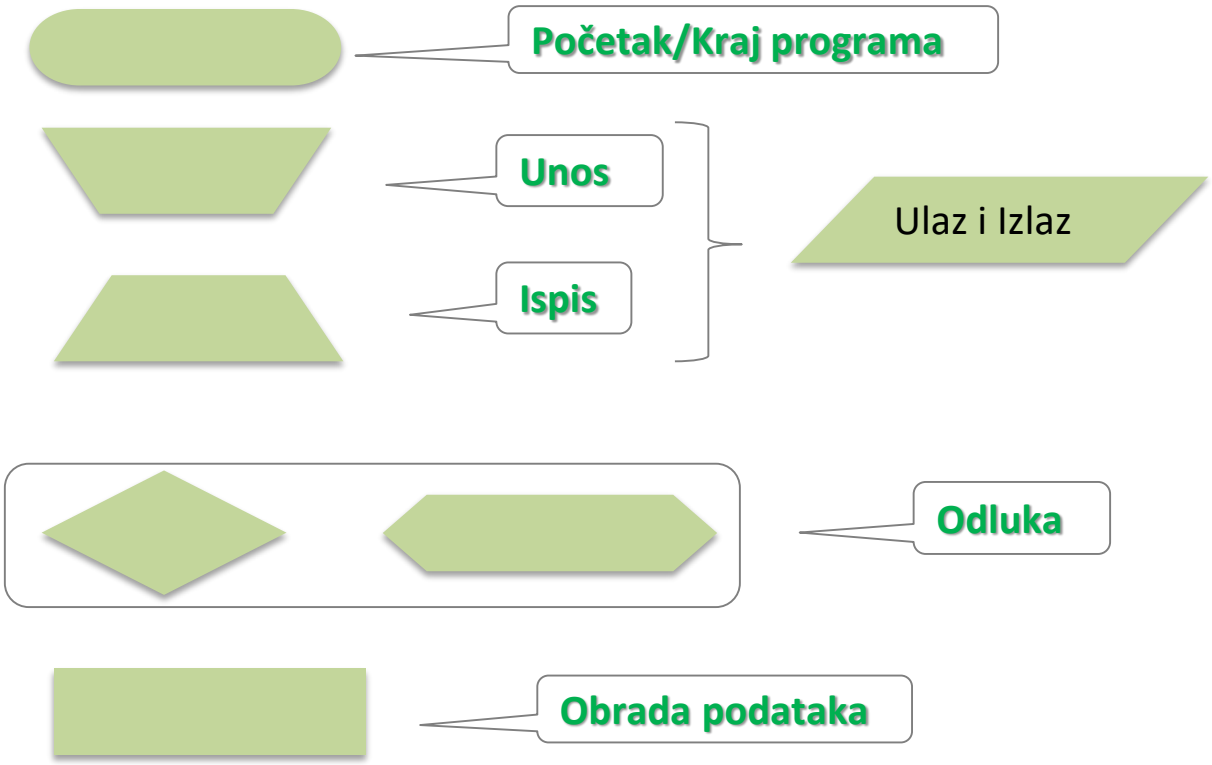
- Kraj profinjavanja ovisi o onome tko definira algoritam, odnosno profinjava se do akcija koje zna izvršiti (profesionalni kuhar vs. početnik u kuhanju).
- Cilj algoritma je cjelokupni zadatak svesti na niz jednostavnih, manjih radnji.

- Pretpostavke za pisanje algoritma:
  - Najprije moramo podrobno upoznati zadatak koji namjeravamo riješiti
  - Dobro poznavati zadatak, znači ponešto znati i o rješenju jer trebamo barem naslućivati što očekujemo od računala
  - Moramo znati kakva će informacija trebati računalu i odakle će je dobiti
  - Moramo predvidjeti kakva će informacija izaći na izlaznoj strani
  - Trebamo napisati postupak prema kojem će teći program – algoritam
  - Postupak mora biti završiv

- Dva algoritma su ekvivalentna kada su:
  - klase ulaznih objekata dovoljne i za jedan i za drugi algoritam iste
  - rezultati i jednog i drugog algoritma za jednake ulazne objekte su jednaki

Dijagram toka i pseudokod

- Algoritam se grafički predočava **dijagramom toka**
- U tom prikazu svaki algoritamski korak je iskazan odgovarajućim grafičkim simbolom.



- Algoritam se tekstualno prikazuje **pseudokodom**.
- Pseudokod:
  - Interpretativni kod - način zapisivanja koji zamjenjuje programski jezik, a koristi se kao pomoć pri oblikovanju programa.
  - Instrukcije u pseudokodu nalik su instrukcijama programskog jezika.
  - Instrukcije pseudokoda pisane su prirodnim jezikom umjesto u programskom jeziku.
  - Prirodni jezik pseudokoda omogućava praćenje problema na lagan način.
- Pseudokôd je kvaziprogram (grč. pseudos – laž) jer premda nalikuje računalnom programu, zapravo nije napisan u programskom jeziku koji bi se mogao izravno primijeniti na bilo kojemu računalu
- Pseudokôd se sastoji od kratkih izraza na govornom jeziku koji opisuju i ukratko objašnjavaju pojedine zadatke algoritma

- Osoba koja piše pseudokôd ne mora znati programski jezik i ne mora razmišljati o pravilima pisanja programskog jezika.

- Pseudokôd bi trebao biti napisan tako da programer može na temelju njega napisati program u bilo kojem programskom jeziku.

- *Primjer algoritma za zbrajanje prirodnih brojeva*
  1. napisati brojeve jedan ispod drugog, tako da znamenke iste težine budu u istom stupcu, tj. poravnato desno
  2. sa zbrajanjem se započinje s desne strane
  3. zbrajaju se znamenke u istom stupcu i zbroju se pribraja eventualni prijenos
  4. prijenos je na početku jednak nuli
  5. ako je zbroj znamenki u stupcu ≤ 9, onda ga zapisujemo i prelazimo na sljedeći stupac
  6. inače, zapisujemo znamenku jedinica, a prijenos postaje znamenka desetica
  7. koraci od 3. do 6. se ponavljaju sve dok ne zbrojimo znamenke u krajnjem lijevom stupcu i zapišemo taj zbroj.

Primjer izrade algoritma – problem čovjeka, kupusa, koze i vuka

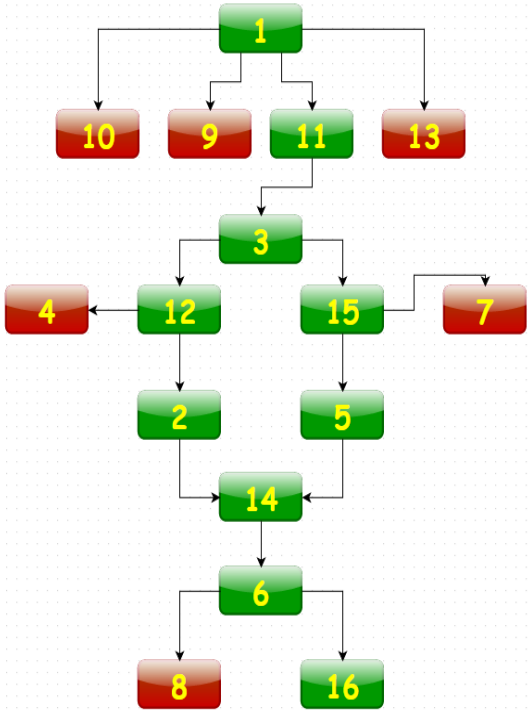
- Čovjek želi s istočne strane rijeke prenijeti kupus, kozu i vuka na zapadnu stranu rijeke.
- U njegov brod može stati samo on i još jedno biće: kupus, koza ili vuk.
- Čovjek ne može ostaviti kozu nasamo s kupusom jer će koza pojesti kupus.
- Isto tako ne može ostaviti nasamo vuka s kozom jer će vuk pojesti kozu.
- Kako će čovjek riješiti problem?

- Što bi bio prikladan način logičkog prikazivanja danog problema? Pošto samo važni aspekti problema trebaju biti uključeni u logički prikaz, sve nevažne informacije se mogu isključiti. Prikaz problema koji uzima u obzir samo bitne informacije se naziva **apstrakcija**.
- U ovom primjeru, da li je boja broda važna informacija, ili širina rijeke, ili čovjekovo ime? Ne, važne informacije su samo gdje se (na kojoj strani rijeke) u kojem trenutku nalaze čovjek, kupus, koza i vuk.

čovjek	kupus	koza	vuk
istok	istok	istok	istok

- Sva moguća stanja ovog problema su prikazana u sljedećoj tablici. Tih stanja ima ukupno 2*2*2*2=16, što odgovara činjenici da svaki od sudionika može biti ili na lijevoj ili na desnoj strani rijeke. Međutim neka od tih stanja nisu dozvoljena jer u njima koza jede kupus ili vuk jede kozu.

	čovjek	kupus	koza	vuk	komentar
1	istok	istok	istok	istok	
2	istok	istok	istok	zapad	
3	istok	istok	zapad	istok	
4	istok	istok	zapad	zapad	Vuk jede kozu
5	istok	zapad	istok	istok	
6	istok	zapad	istok	zapad	
7	istok	zapad	zapad	istok	Koza jede kupus
8	istok	zapad	zapad	zapad	Vuk jede kozu ili Koza jede kupus
9	zapad	istok	istok	istok	Vuk jede kozu ili Koza jede kupus
10	zapad	istok	istok	zapad	Koza jede kupus
11	zapad	istok	zapad	istok	
12	zapad	istok	zapad	zapad	
13	zapad	zapad	istok	istok	Vuk jede kozu
14	zapad	zapad	istok	zapad	
15	zapad	zapad	zapad	istok	
16	zapad	zapad	zapad	zapad	



Algoritamske strukture

- Algoritamska struktura opisuje način i redoslijed izvršavanja pojedinih radnji koje dovode do konačnog rješenja zadatka.

- Razlikuje se nekoliko osnovnih algoritamskih struktura:
  - Linijska struktura
  - Razgranata struktura
  - Ciklička struktura