# Algoritam

## 1. DEFINICIJA ALGORITMA

Algoritam jeste jedan od osnovnih pojmova matematike i računarstva. Prve algoritme srećemo kod starih Grka. Svima su poznati Euklidov algoritam i Eratostenovo sito (250. g. p. n.e.).

Algoritam predstavlja uputstvo za rešavanje nekog zadatka u cilju dobijanja rešenja (posle konačno mnogo vremena).

Pri rešavanju nekog složenog zadatka (problema) korsitimo rastavljanje (dekompoziciju) zadatka na prostije podzadatke (podprobleme). Postupak možemo nastaviti sve dok se ne dobije jedan konačan skup relativno jednostavnih podzadataka. Takve podzadatke nazvaćemo elementarnim koracima ili elementarnim operacijama. Svaki takav korak definiše koju operaciju i kojim redosledom se takva obrada izvršava u cilju dobijanja rešenja datog problema. Ukoliko je elementarni korak relativno složen tada se on može zadati u vidu uputstva (pravila) za rešavanje takvog podproblema. Zato elementarne korake nazivamo i (elementarnim) pravilima.

Opis toka odvijanja elementarnih operacija (koraka) u cilju rešavanja nekog zadatka (problema) naziva se procedura. Procedura se sastoji od konačno mnogo elementarnih koraka koji se mogu mehanički izvršavati u određenom strogo definisanom redosledu i to za konačno vreme.

**Def. 1: Algoritam je konačan (uređen) skup strogo definisanih algoritamskih koraka (pravila) čijom primenom na ulazne podatke (i međurezultate) dobijamo rešenje zadatka posle konačno mnogo vremena.**

## 2. OSOBINE ALGORITAMA

**1.Diskretnost.**

Svaki algoritam predstavlja konačan uređen skup algoritamskih koraka. To je rezultat dekompozicije (diskretizacije) problema koji se rešava od strane čoveka. Pri tome, algoritam nije skup algoritamskih koraka, već uređen skup ili niz, jer je bitan redosled algoritamskih koraka u zapisu algoritma. Takođe kažmo da je proces izvršenja algoritma diskretan u vremenu. To znači da se u svakom vremenskom trenutku (intervalu) izvršava samo jedan algoritamski korak. Proces izvršavanja algoritma jeste konačan niz algoritamskih koraka koji se po strogo definisanom redosledu izvršavaju. To je slučaj kada imamo jednog izvršioca algoritma (jedan procesor). Postoje i paralelni algoritmi, koji omogućuju da se dva ili više algoritamskih koraka izvršavaju istovremeno.

**2. Determinisanost (određenost).**

Svaki algoritamski korak treba da je definisan jasno, strogo (tačno) i nedvosmisleno. Tumačenje i izvršavanje pravila algoritma ne sme zavisiti od volje čoveka ili mašine. Posle izvršenja nekog algoritamskog koraka strogo je definisan prelaz na sledeći algoritamski korak. To znači da je izvršenje algoritma deterministički proces i da se može automatski izvršavati. Opis algoritma na prirodnom jeziku može dovesti do dvosmislenosti.

**3. Izvršivost.**

Uspešnu definiciju izvršivosti dao je D. Knut. On kaže da je algoritamski korak izvršiv ako je čovek u stanju da ga izvrši za konačno vreme (pomoću olovke i papira). Kod algoritamski rešivih zadataka nema neizvršivih koraka. Međutim, moguće je lako formulisati  sadrži 7 uzastopnih devetki tada$\pi$pravilo koje nije izvršivo. Na primer: Ako razvoj broja  sabrati sve preostale cifre.

**4. Konačnost.**

Osobina konačnosti algoritma jeste zahtev da se izvršenje svakog algoritma završi posle konačno mnogo vremena. Drugim rečima,izvršenje svakog algoritma je postignuto posle konačno mnogo primena algoritamskih koraka. Zbog osobina 1. i 3. to znači da svaki algoritamski korak ma kog algoritma mora da se izvrši konačno mnogo puta. U suprotnom, nemamo algoritam. Takav je sledeći primer procedure koja se nikad ne završava : Korak 1 : Neka je i = 0 ; Korak 2 : Neka i dobije vrednost i +1 ; Korak 3 : Pređi na korak 2 . Iako imamo konačno mnogo algoritamskih koraka, pri čemu je svaki od njih strogo definisan i izvršiv, ovo nije algoritam. Kod složenijih algoritama (zadataka) javlja se potreba formalnog dokaza konačnosti algoritma.

**5. Ulaz i izlaz algoritma.**

Svaki algoritam ima dva posebno izdvojena (konačna) skupa podataka (veličina). Prva jeste skup ulaznih a druga skup izlaznih veličina. Broj veličina u ovim skupovima može biti i nula. Skup ulaznih veličina algoritma predstavlja polazne veličine (podatke) zadatka koji se rešava. Skup izlaznih veličina jeste traženo rešenje (rezultat) postavljenog zadatka. Ukratko, ove skupove nazivamo ulaz i izlaz algoritma. (Za podatke koji na pripadaju skupu ulaznih podataka algoritma kažemo da algoritam nije primenljiv).

**6. Masovnost (Univerzalnost).**

Univerzalnost je osobina algoritma da se može primeniti na što širu klasu problema. To upravo znači da ulazne veličine algoritma mogu uzimati početne vrednosti iz što obilnijih (masovnijih)

skupova podataka. Algoritam jeste opšte uputstvo koje se može primeniti na ma koji izbor vrednosti ulaznih veličina. Zbog ove osobine,možemo reći da je algoritam bolji ukoliko je univerzalniji (termin masovnost manje odgovara ovoj osobini).

**7. Elementarnost algoritamskih koraka.**

Algoritam treba da sadrži algoritamske korake koji predstavljaju elementarne operacije koje korisnik algoritma može da razume ili izvršilac algoritma da izvrši. Za potrebe čoveka, algoritamski koraci mogu biti kompleksnije fundamentalne ili logičke celine u složenom algoritmu. Međutim, za potrebe pisanja programa, algoritam sadrži elementarne algoritamske korake koji odgovaraju naredbama ili pozivima potprograma, algoritam sadži elementarne algoritamske korake koji odgovaraju naredbama ili pozivima potprograma programskog jezika.

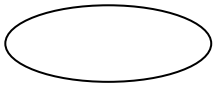
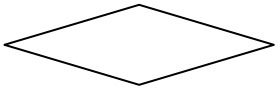
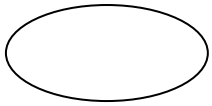
**8. Rezultativnost (usmerenost).**

Algoritam je tako definisan da polazeći od proizvoljnih vrednosti ulaznih veličina primena algoritamskih koraka vodi (usmerava) strogo ka dobijanju traženog rezultata.

# 3. ZAPIS ALGORITMA

## 3.1. Grafički zapis

Jedan od najjednostavnijih i često korišćenih načina zapisa algoritama jeste dijagram toka (ili blok šema) algoritma (tokovnik). Dijagram toka predstavlja jednu varijantu grafičkog opisa algoritma. Svaki algoritamski korak predstavljen je grafičkim simbolom (blokom). Svi blokovi su povezani linijama sa mogućim strelicama. Na taj način se zadaje struktura algoritma i redosled izvršavanja algoritamskih koraka. Oblik grafičkog simbola ukazuje na vrstu algoritamskog koraka,odnosno njegovu funkciju u algoritmu. U tabeli su dati najčešće korišćeni grafički simboli i njihova funkcija.

| **Grafički simbol algoritamskog koraka** | **Funkcija algoritamskog koraka** |
|---|---|
|  | Početak algoritma |
|  | Blok za unos podataka |
|  | Blok izračunavanja |
|  | Uslovni blok |
|  | Blok izdavanja podataka |
|  | Blok kraja |

## 3.2. Zapis algoritama skupom pravila

Najprirodniji način zapisa algoritama jeste pomoću prirodnog jezika kojim komuniciraju ljudi. Tako se algoritam može saopštiti govorom ili tekstualno. Primeri za to su razni kulinarski recepti, uputstva za rukovanje mašinama, koja jesu neka vrsta algoritama. Takvi opisi mogu biti neprecizni, pa čak i dvosmisleni zbog složenosti prirodnog jezika. Međutim, zapis algoritama pomoću skupa pravila ima zadatak da pregledno, jasno i tačno opiše algoritamske korake pomoću reči, simbola i rečenica prirodnog jezika. Oblik pravila algoritma je na određen način precizan tako da su ona jasna i razumljiva svakome.

U zapisu algoritama skupom pravila može se uvesti ime algoritma. Pored toga algoritamska pravila se najčešće označavaju rednim brojevima (0,1,2,. . . ), sa dodatnom simboličkom oznakom u vidu slova ili reči. Izvršenje algoritma počinje od pravila sa rednim brojem 0 ili 1. Zatim se pravila algoritma izvršavaju prirodnim redosledom (redno) dok se ne dođe do pravila za grananje,prelaz ili pravila za cikličko izvršavanje algoritamskih koraka.

Navodimo neka najčešće upotrebljavana pravila za zapis algoritma :

**1.Pravilo dodeljivanja :**

:= Izračunata vrednost izraza dodeljuje se promenljivoj sa leve strane simbola dodeljivanja := . .⇐ , ←Alternativni simboli mogu biti = ,

**2. Pravilo uslovnog grananja :**

if then Ako je ispunjen tada se izvršava ,a u suprotnom prelazi se na sledeće pravilo.

**3.Pravilo grananja :**

if then else Ako je ispunjen izvršava se ,a u suprotnom .

**4.Pravilo bezuslovnog grananja (skoka) :**

go to Ovo pravilo ukazuje da se prekida prirodni redosled izvršenja pravila algoritma i zahteva prelaz na pravilo označeno sa .

**5.Pravilo za prekid izvršenja algoritma :**

Kraj (Stop) Posle izvršenja ovog pravila prekida se izvršenje algoritma.

**6. Pravilo za ulaz podataka :**

Ulaz ()

**7. Pravilo za izlaz podataka :**

Izlaz ()

**8. Pravilo ciklusa :**

Ponavljaj . . . Sve dok se ne ispuni Ovo pravilo definiše ponovljeno izvršavanje niza pravila ,. . . , sve dok nije ispunjen. Kada se ispuni prelazi se na sledeće pravilo.

**Primer Euklidovog algoritma za nalaženje najvećeg zajedničkog delioca.**

E0. Euklid

E1. Ulaz (m,n) }ostatak deljenja{

E2. r:= m MOD n

E3. Ako r=0 tada pređi na E7

E4. m:=n

E5. n:=r

E6. Pređi na E2

E7. nzd:=n

E8. Izlaz (nzd)

E9. Kraj Slična algoritamska pravila imamo kod opisa algoritama pomoću pseudo koda.

Pseudo kod koristi pravila bliska naredbama programskog jezika. Pravila su jednostavna, čitljiva i nedvosmislena.

# 4.ELEMENTARNE ALGORITAMSKE STRUKTURE

Postoje tri elementarne algoritamske strukture:

1. Linijska - sve akcije se izvršavaju ta čno jednom u redosledu u kome su navedene
2. Razgranata (SELEKCIJA) - omogu ćuje da se od više grupa akcija, koje se nalaze u razli čitim granama razgranate strukture, izabere ona koja će se izvršiti jednom, dok se sve ostale grupe akcija ne će izvršitini nijednom
3. Ciklična (ITERACIJA) - skup akcija može se izvršiti više puta

Algoritamsko rešenje bilo kog problema može se uvek zapisati koriš ćenjem samo ove tri strukture