ALGORITMI
(pojam, strukture, kodiranje i programske strukture)

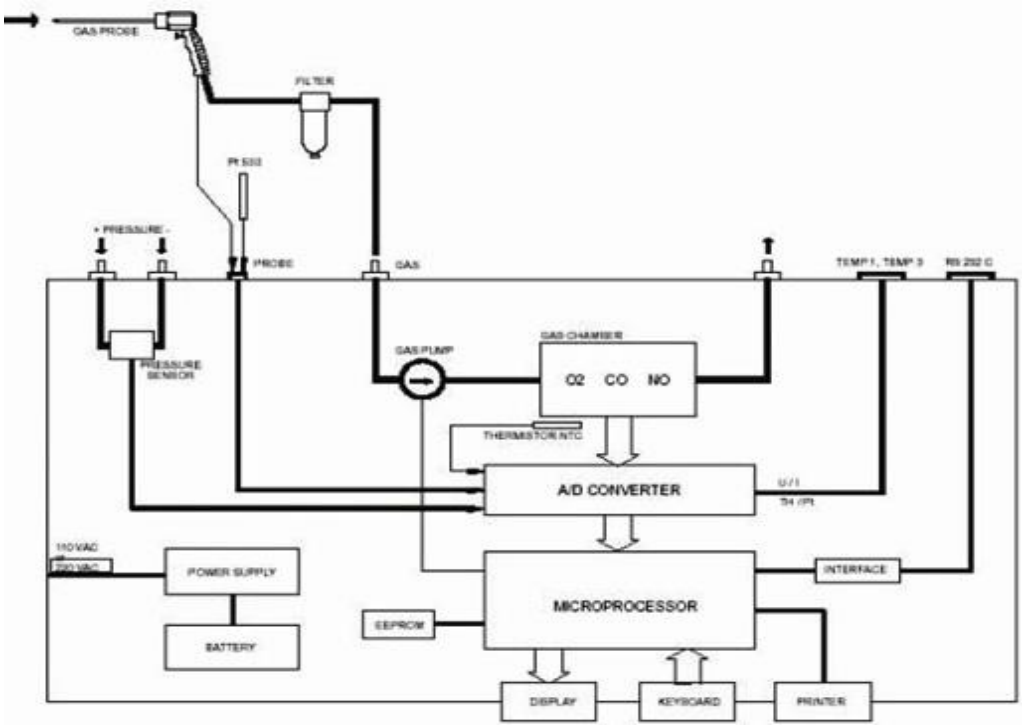
Algoritam predstavlja skup akcija sa definiranim redosledom njihovog obavljanja, koji primenjen na polazni skup podataka, dovodi do traženih rezultata. U procesu programiranja, skup akcija definiran je mogućnostima računala, odnosno naredbama programskog jezika koji se koristi, dok se redosled izvršavanja akcija zadaje pomoću algoritamskih (programskih) struktura.

ALGORITAM

Algoritam je skup pravila ili pravilo sa svojstvom preciznosti, jednoznačnosti te obuhvata konačan broj koraka, a svaki korak je opisan instrukcijom. Instrukcije moraju biti izvedive i jednoznačne. Algoritam opisuje rešavanje nekoga problema. Postupak obavljanja algoritma je algoritamski proces. Algoritam ima definirane početne objekte nad kojima se obavljaju operacije, a ishod toga je skup rezultata tj. završnih objekata i on je delotvoran. Da bi algoritam bio učinkovit rezultat se mora dobiti u prihvatljivom ili razumnom vremenu. Instrukcije se mogu izvršiti nekoliko puta te instrukcije moraju pokazivati na ponavljanje, ali za bilo koju vrednost ulaznih podataka algoritam završava nakon konačnog broja ponavljanja. Kod zapisivanja algoritama upotrebljava se programski jezik C, reč je o nedovršenom kodu gde su neki nizovi naredbi zamenjeni tekstom. Analiza algoritma podrazumeva procenu vremena za izvršavanje toga algoritma, a vreme se poistovećuje sa brojem operacija koje odgovarajući program treba obaviti i on se izražava kao funkcija. Algoritam se zapisuje u :

• Obliku pseudo jezika (govornog jezika koji oponaša programski jezik)

• Grafičkom obliku tzv. Blok dijagram ili dijagram toka programa



Slika: BLOK DIJAGRAM

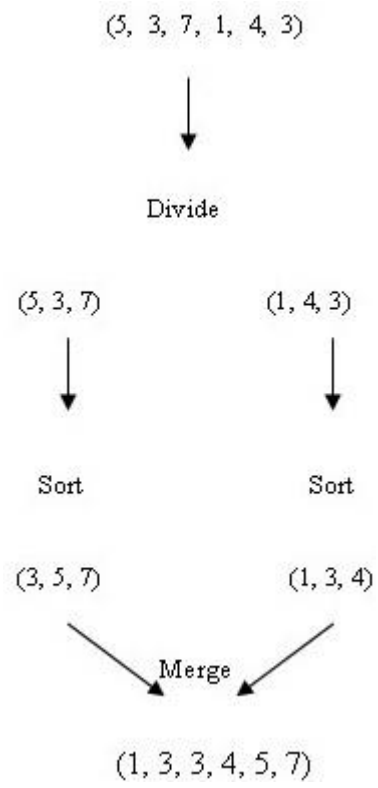
OBLIKOVANJE ALGORITMA

Oblikovanje algoritama se deli na tehnike: podeli pa vladaj, dinamičko programiranje, pohlepni pristup i „backtracking“. Svaka od ovih metoda ne garantira tačno rešenje problema i zbog toga se uvek treba napraviti provera.

PODIJELI PA VLADAJ

Metoda podeli pa vladaj se deli na tri primera: sortiranje sažimanjem, traženjem elemenata u listi i množenje dugačkih cijelih brojeva. Algoritam za sortiranjem liste se može tumačiti da što je lista dulja to ju je teže sortirati, velika sortirana lista se dobiva relativno jednostavnim postupkom sažimanja malih sortiranih lista.

Primer 1.
Sortiranje sažimanjem

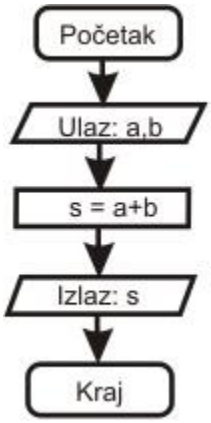


PROGRAMSKE STRUKTURE

Postoje tri programske strukture a to su:

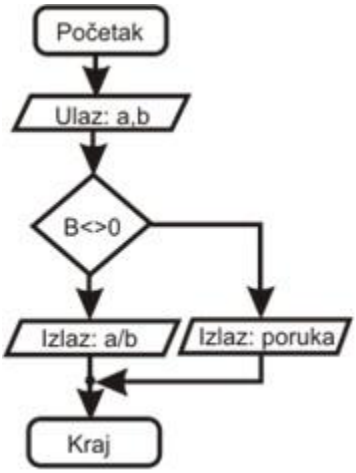
1. **Linijska** (sled)
2. **Razgranata** (selekcija)
3. **Ciklička** (iteracija)

1. Linijska (sled) programska struktura: sve akcije se izvršavaju tačno jednom u redosledu u kome su navedene.



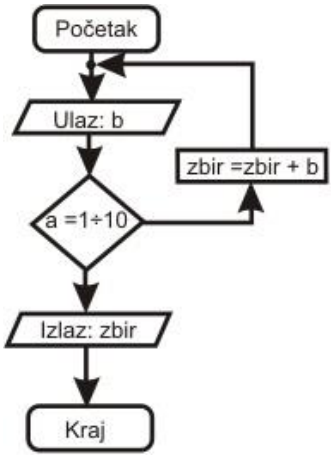
SHEMA: Linijska programska struktura

2. Razgranata (selekcija) programska struktura: omogućava 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 izvršiti ni jednom.



Shema: Razgranata algoritamska struktura

3.Ciklička (iteracija) programska struktura: skup akcija može se izvršiti više puta.



Shema: Ciklička programska struktura

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

ALGORITAMSKE STRUKTURE

1. **Sledbena** (linearne ili sekvencijalne)

• *Početak i kraj*

• *Definiranje varijabli i konstanti*

• *Ulaz*

• *Izlaz*

• *Aritmetičke i logičke operacije*

2. **Struktura bezuslovnog skoka**
3. **Struktura grananja** (sadrži logičke operacije) kombinira se sa:
   - *Sledbenom strukturom*
   - *Strukturom bezuslovnog skoka*

4. **Struktura iteracije** (ponavljanja ili petlje)

STRUKTURA BEZUVJETNOG SKOKA (narušavanje linearnosti)

- Koristi se za testiranje algoritma (preskače deo algoritma)
- Izaziva grešku bezuslovnog ponavljanja (tzv. Beskonačna petlja ili iteracija)
- Kombinira se s strukturom grananja radi narušavanja linearnosti / uspostavljanja ponavljanja (dela) algoritma.

PSEUDO JEZIK	BLOK DIJAGRAM
	∫
x. Idi na y	
Gde su x i y brojevi linija algoritma	bez obzira na smer
Struktura bezuslovnog skoka	

SLOŽENE ALGORITAMSKE STRUKTURE

Složene algoritamske strukture nastaju kada se u elementarnim strukturama pojedini algoritamski koraci zamene drugim algoritamskim koracima ili dugim elementarnim strukturama. Tako se na osnovama navedenog principa nadgradnje mogu dati pravila za dobijanje složenih algoritamskih struktura.

To se postiže:

1) kada se u prostoj linijskoj strukturi u sastavu razgranate linijske ili ciklične strukture algoritamski korak prve vrste (ne dovodi do grananja algoritma i ima jedan ulaz i jedan izlaz) zameni algoritamskim korakom druge vrste (dovodi do grananja algoritma i ima jedan ulaz i više izlaza)

2) kada se u prostoj linijskoj strukturi u sastavu razgranate linijske ili ciklične strukture algoritamski korak prve vrste zameni novom razgranatom linijskom ili cikličnom strukturom;

3) zamenom algoritamskog koraka proizvoljne elementarne strukture algoritamskim korakom za poziv podprograma

4) kada se u već formiranim algoritamskim strukturama ponavljaju zamene iz pravila datih u tačkama (1), (2) i (3).

Primenom navedenih pravila mogu se formirati najraznovrsnije složene algoritamske strukture.

Jedna od mogućih klasifikacija ovih struktura je:

- *složene razgranate linijske strukture*
- *složene ciklične strukture*
- *složene razgranate i ciklične strukture*
- *algoritamske strukture sa potprogramom*

Napomena: Za složene algoritamske strukture dobijene principom nadgradnje se može postaviti i zahtev da budu strukturirane (da imaju jedan ulaz i jedan izlaz) čime se zadovoljava princip strukturiranosti primenom tog zahteva u većini slučajeva algoritamske strukture postaju logički preglednije i pogodnije za programsku realizaciju.

KANONSKE I NEKANONSKE ALGORITAMSKE STRUKTURE

Kombinacijom elementarnih struktura formiraju se kanonske, kvazi-kanonske i nekanonske algoritamske strukture.

Postoje 4 kanonske, 5 kvazi-kanonskih i 2 nekanonske algoritamske strukture a to su :

1. **kanonske algoritamske strukture**
   - *kanonska linijska struktura*
   - *kanonska razgranata struktura*
   - *kanonska ciklička struktura*
2. **kvazi- kanonske algoritamske strukture**
3. **nekanonske algoritamske strukture**

TESTIRANJE ALGORITMA

Izvodi se kao i u matematici uvrštavanjem vrednosti u algoritam. Algoritam se testira sekvencijalno praćenjem svakog reda (instrukcije) algoritma od početka do kraja, uz zapisivanje vrednosti koje varijable usput poprimaju, da bi se u konačnici i saznala konačna vrednost izlaznih varijabli.

Pr. zadatka:

Kolika je vrednost varijable s nakon izvođenja algoritma, ako za x učitamo 2, a za y učitamo 5:

1. X=0, Y=0, S=0
2. Učitaj X
3. Učitaj Y
4. S=S+Y
5. Ispiši S S = 5

KODIRANJE

Algoritam započinje s praznim rečnikom, no uzmimo neki trenutak u toku kodiranja, kad rečnik već sadrži neke stringove. Analiziramo prefiks koji sledi u ulaznoj struji, počevši sa praznim prefiksom. Ako njemu odgovarajući string (prefiks + sljedeći znak, odnosno P+Z) postoji u rečniku, proširujemo prefiks P znakom Z. Proširenje ponavljamo sve dok ne dobijemo string koji ne postoji u rečniku. Tada na izlaz pošaljemo kodnu reč koja odgovara P-u, a zatim i znak Z. Sljedeći prefiks koji analiziramo direktno se nastavlja na obrađeni. Poseban je slučaj ako u rečniku ne postoji ni početni string od samo jednog znaka (to se uvek događa na početku kodiranja). Tada se na izlaz šalje posebna kodna reč koja označava prazni string. Iza nje se šalje dotični znak i taj se znak dodaje u rečnik. Izlaz iz algoritma su parovi kodna reč-znak (K,Z). Svaki put kad se taj par pošalje na izlaz, string iz rečnika koji odgovara kodnoj reči K proširi se znakom Z i taj novi string se doda u rečnik. To znači da u trenutku kad dodajemo neki string, u rečniku već postoje svi stringovi dobiveni oduzimanjem znakova s kraja tog stringa.

Algoritam za kodiranje

1. Na početku su rečnik i P prazni;
2. Z := sljedeći znak u ulaznoj struji;
3. Postoji li string P+Z u rečniku?
   a. ako postoji, P := P+Z (P proširi Z-om);
   b. ako ne postoji,
   --u izlaznu struju pošalji dva podatka, i to:
   - kodnu reč stringa jednakog P-u (ako je P prazan, šalji nulu);
   - Z, u istom obliku kao što je učitan iz ulazne struje;
   --na prvo prazno mesto u rečniku zapiši string P+Z;
   --isprazni P;
   c. ima li još znakova u ulaznoj struji?
   --ako ima, vrati se na korak 2;
   --ako nema:
   - ako P nije prazan, u izlaznu struju pošalji kodnu reč stringa jednakog P-u;
   - KRAJ

ZAKLJUČAK

Algoritamske strukture važan deo programiranja, pomoću njih se zadaje redosled izvršavanja akcija, tj. problema. One su bitan deo algoritama, jer se algoritmi nikako ne mogli strukturisati, tj. rešavati. Algoritam je postupak kojim se u konačnom broju koraka i u konačnom vremenu dolazi do rešenja problema ili spoznaje da rešenja nema. Svaka vrsta stroja za obradu podataka ima svoj jezik i ne može se koristiti na drugom stroju, čak ni na različitim strojevima istog proizvođača.