Uvod

Kompjuterski programi

Rešavanje problema korišćenjem kompjuterskih programa se može razložiti na više etapa:

Definisanje problema je postupak u kome naručilac i programer na prirodnom jeziku (srpski, engleski, ...) definišu koje probleme program treba da rešava. Iako nije neophodno, poželjno je da naručilac ima osnovno informatičko znanje, kako bi se lakše sporazumeo sa programerom i kako bi se izbegle eventualne greške usled različitog tumačenja istog problema.

Analiza problema obuhvata definisanje ulaznih i izlaznih podataka, moguća ograničenja njihovih vrednosti, kao i matematički model koji će biti korišćen za rešavanje datog problema.

Definisanje algoritma koji rešava problem, što podrazumeva definisanje konačnog uređenog niza pravila kojima se rešava određeni tip problema.

Projektovanje programa podrazumeva izbor platforme i programskog jezika, a zatim i definisanje arhitekture samog programa i načina čuvanja podataka.

Kodiranje je prevođenje pravila definisanih algoritmom na konkretan programski jezik. Dobro definisan algoritam znatno olakšava pisanje programa.

Testiranje je faza u razvoju kompjuterskih programa koja treba da osigura blagovremeno otkrivanje i uklanjanje grešaka. Testovi pomoću kojih se ispituje funkcionalnost programa treba da obuhvate sve opsege ulaznih promenljivih, kao i sve moguće grane u izvršenju programa. Pored kontrole izvršenja programa potrebno je izvršiti i testiranje robustnosti programa u slučajevima unosa neodgovarajućih podataka od strane korisnika.

Analiza rezultata podrazumeva poređenje dobijenih rezultata sa teorijskim ili eksperimentalnim rezultatima, kao i modifikaciju modela u slučajevima kada dobijeni rezultati nisu u granicama dozvoljene tolerancije.

Isporuka programa je proces u kome se program putem različitih medija (disketa, CD, FTP, internet, ...) stavlja na raspolaganje naručiocu, tako da ga on može samostalno koristiti.

Održavanje programa je dugotrajan proces, koji podrazumeva obuku korisnika, ispravku uočenih nedostataka i prilagođavanje programa zahtevima korisnika.

U slučaju razvoja velikih programskih rešenja postoje različiti tipovi ciklusa kroz koje prolazi svaki program. Izbor najpogodnijeg tipa zavisi od namene programa, broja učesnika na projektu, strategije kompanije koja se bavi razvojem i mnogih drugih parametara.

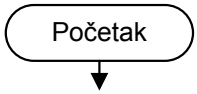
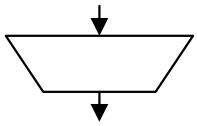
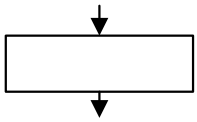
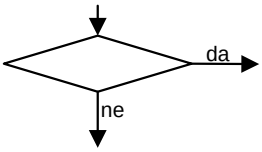
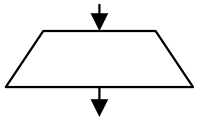
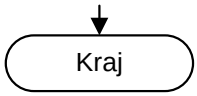
Algoritmi

Algoritam je konačan uređen niz precizno formulisanih pravila kojima se rešava jedan ili čitava klasa problema.

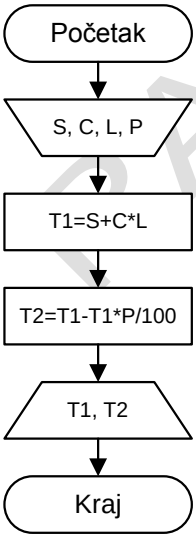
Svaki algoritam se može opisati tekstulano na prirodnom jeziku (srpski, engleski, ...). Na primer, algoritam za računanje cene taksi usluge bi mogao tekstualno da se opiše na sledeći način:

1. Definisati ulazne veličine: cenu "starta" S, cenu po pređenom kilometru C, broj pređenih kilometara L i popust u procentima P.
2. Izračunati cenu bez popusta T1 kao T1=S+C*L.
3. Izračunati cenu sa popustom T2 kao T2=T1-T1*P/100.
4. Prikazati izlazne veličine: cenu bez popusta T1 i cenu sa popustom T2.

S obzirom da je ovakav način opisivanja algoritma teško čitljiv u slučajevima složenijih problema, znatno češće se koristi grafički prikaz pomoću takozvane **algoritamske šeme**. Algoritamska šema se može sastojati od sledećih grafičkih simbola:

Grafički simbol	Značenje
	Početak algoritma
	Ulazne veličine
	Izračunavanje
	Uslovno grananje
	Izlazne veličine
	Kraj algoritma

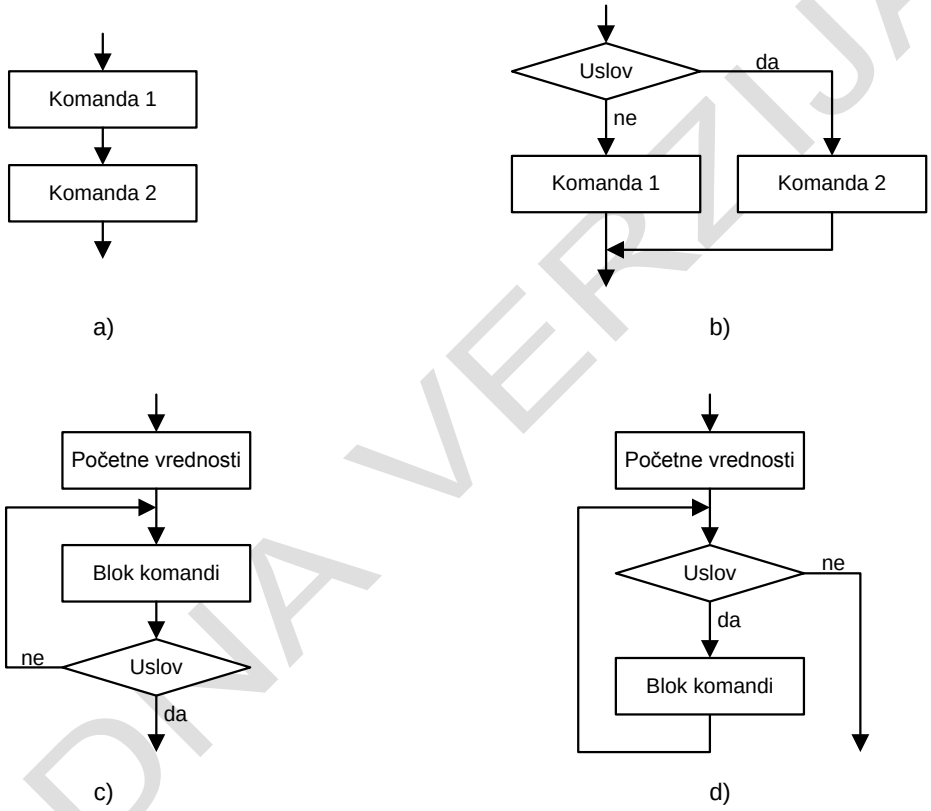
Korišćenjem navedenih simbola moguće je prikazati prethodno definisani algoritam za izračunavanje cene taksi usluge na sledeći način:



Slika ### Algoritamska šema za izračunavanje cene taksi usluge

Svaki algoritam se, bez obzira na stepen njegove složenosti, može predstaviti korišćenjem tri osnovne strukure:

- **Linijska** struktura (Slika ###a) podrazumeva izvršavanje komandi jedne za drugom.
- **Razgranata** struktura (Slika ###b) omogućava grananje izvršenja programa u zavisnosti od ispunjenosti odgovarajućeg uslova.
- **Ciklična** struktura (Slike ###a i ###b) predstavlja niz komandi koje se mogu ponavljati više puta u zavisnosti od ispunjenosti odgovarajućeg uslova. U slučaju kada se uslov nalazi na kraju ciklusa (Slika ###c), blok komandi se izvršava bar jednom, dok se u slučaju kada je uslov na početku ciklusa (Slika ###d) blok naredbi ne mora izvršiti ni jednom.



Slika ### Osnovne algoritamske strukture: **a)** Linijska; **b)** Razgranata; **c)** Ciklična sa uslovom na kraju; **d)** Ciklična sa uslovom na početku

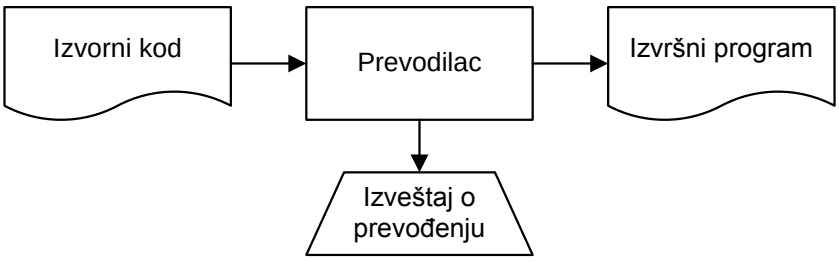
Vrste programskih jezika

Sve programske jezike možemo podeliti na više kategorija u zavisnosti od kriterijuma na osnovu koga se podela vrši:

Prema stepenu bliskosti sa arhitekturom računara programske jezike možemo podeliti na jezike niskog nivoa (mašinski jezik, Assembler) i jezike visokog nivoa (FORTRAN, Pascal, C, C++, Java, C#). Programski jezici niskog nivoa podrazumevaju bliskost sa hardverom, dok jezici višeg nivoa ne zavise od vrste hardvera, već se, zahvaljujući odgovarajućem prevodiocu (kompajleru), mogu koristiti na bilo kom računaru.

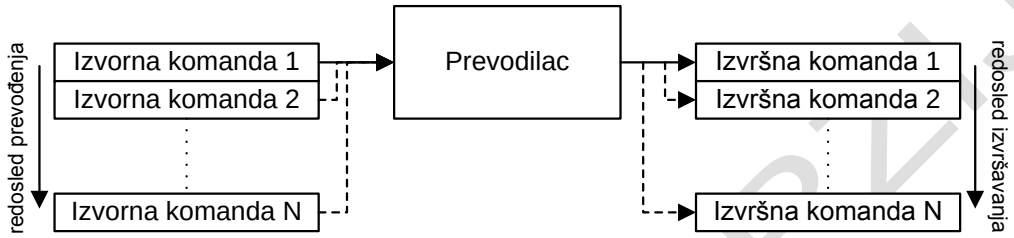
Prema načinu prevođenja programa programske jezike možemo podeliti na **kompajlerske** i **interpretatorske**.

U slučaju kompajlerskih jezika prevođenje kompletnog koda se vrši pre izvršenja programa, a dobijeni program se može kasnije izvršavati bez postojanja kompajlera.



Slika ### Šematski prikaz procesa prevođenja kompajlerskih jezika

Sa druge strane, kod interpretatorskih jezika se prevođenje svake komande vrši neposredno pre njenog izvršenja, što znatno usporava rad programa i zahteva postojanje kompajlera u memoriji za vreme izvršavanja programa.



Slika ### Šematski prikaz prevođenja interpretatorskih jezika

Prema oblasti primene programski jezici se mogu svrstati u različite kategorije:

Programski jezik	Oblast primene
FORTRAN	naučno-tehnički proračuni
Cobol	komercijalni problemi
Lisp	obrada simbola
Modula, Ada	programiranje aplikacija za rad u realnom vremenu
Basic	jezik opšte namene
Pascal	jezik opšte namene pogodan za obučavanje
C	sistemsko programiranje
C++	objektno-orijentisan C
Java i C#	objektni jezici za razvoj složenih i robustnih programa

C

Programski jezik C je kreiran 1972. godine za potrebene operativnog sistema UNIX. Njegov autor je Denis Riči, koji je bio jedan od vodećih projekanata za operativni sistem UNIX.

Prve verzije UNIX-a su bile pisane koristeći asembler, čiji je kod bio veoma veliki i težak za održavanje. Iz tog razloga je nastao jezik C, koji je zadržao efikasnost asemblera, a pružio komfor već postojećih viših programskih jezika, poput Paskala, Kobola i dr. Programski jezik C koristi princip strukturnog programiranja koji omogućava razlaganje problema na manje celine od kojih svaka sme imati samo jedan ulaz i jedan izlaz, izbegavajući korišćenje naredbe GO TO. Na ovaj način informacije se prenose iz bloka u blok sve dok se ne dođe do konačnog rešenja.

Programski jezik C je postao jedan od najrasprostranjenijih programskih jezika svih vremena. Takođe poslužio je kao inspiracija za mnoge druge programske jezike poput C++, C#, Objective-C, Java, PHP, Python, Javascript i dr.