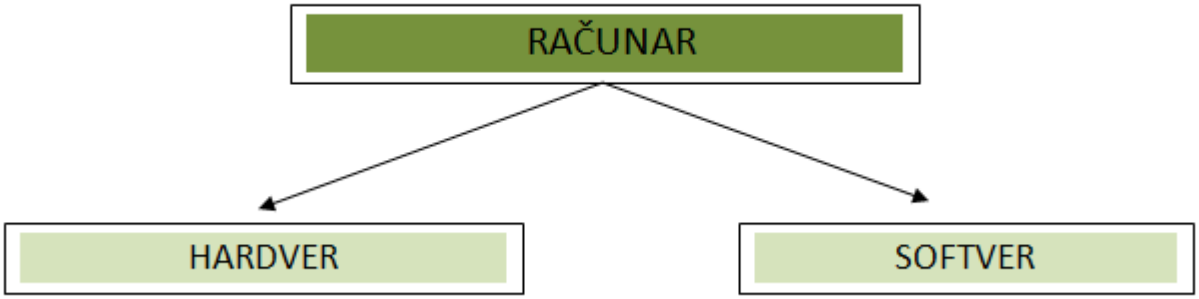
Programski jezici

Hardver (hardware) je fizički delovi računara.

Softver (software) čini sva programska podrška koja se nalazi u memoriji računara.

- svi programi koji se mogu koristiti na nekom računarskom sistemu
- omogućavaju pravilno i efikasno funkcionisanje hardvera



Program - skup naredbi koje računar izvršava da bi obavio određeni posao.
Programi (softver) se mogu podeliti na: sistemski i aplikativni.

Sistemski programi - orijentisani ka radu samog računara.
Aplikativni programi – korisnički programi.

Operativni sistem - skup manjih programa koji koordinisano kontrolišu hardver računara starajući se da računar radi na tačno određeni način, to je posrednik između hardvera računara i ostalih programa koji će se na njemu izvršavati.

Drajveri - važni sistemski programi koji predstavljaju vezu između hardvera i operativnog sistema.

Mašinski jezik

- interni jezik računara sagrađen nad binarnom azbukom
- osnovna prednost je brže izvršavanje od programa napisanih na nekom drugom programskom jeziku

Simbolički (asemblerski) jezici :

- nastali iz mašinskih zamenom koda naredbe odgovarajućim simboličkim oznakama
- binarne adrese operanada zamenjene simboličkim oznakama
- program se ne može direktno izvršavati već se mora prevesti na mašinski jezik
- program koji vrši prevođenje naziva se prevodilac ili asembler

Kompajler

- prevodilac čiji je ulazni jezik viši programski jezik, a izlazni jezik mašinski jezik
- kompajler prevodi na interni mašinski jezik čitav program pa ga izvršava

Interpreter

- prevodilac čiji je ulazni jezik viši programski jezik, izlazni jezik neka posredna forma koja se izvršava
- prevodi jednu naredbu na niz instrukcija mašinskog jezika, izvršava ih,
- prelazi na prevođenje i izvršavanje sledeće naredbe, ...

Linker

- rezultat prevođenja se dobija u formi koja se naziva objektni modul
- da bi se program izvršavao mora se dovesti u izvršnu formu i smestiti u operativnu memoriju
- ovaj posao radi povezivač ili editor veza

Proceduralno orijentisani jezici

- opisuju proceduru ili algoritam za rešavanje problema na način razumljiv računaru
- glavne odlike : lakoća praćenja i razumevanja programa, lakše učenje programskog jezika, lako otklanjanje grešaka, ...

Problemski orijentisani jezici

- opisuju problem koji treba rešiti i potrebne podatke
- opis algoritma je ugrađen u prevodilac (generator)
- specijalizovani za rešavanje uske klase problema

Jezik je skup pravila za komunikaciju između korisnika. Pomoću jezika se predstavljaju i prenose informacije.

Sintaksa — skup pravila za određivanje pravilnih konstrukcija jezika

Semantika — određuje značenje sintaksno ispravnih konstrukcija jezika

Jezike možemo podeliti na:

- prirodne
- veštačke

Prirodni jezik se koristi za komunikaciju između ljudi u pisanoj i govornoj formi.

Programski jezik je veštački jezik koji prvenstveno služi za komunikaciju između čoveka i računara, mada se ponekad programski jezik koristi i za komunikaciju između ljudi). Programski jezik služi za opis programa. Preko programa se obavlja komunikacija između čoveka i računara.

Poželjne osobine programskih jezika su:

- jednostavnost
- izražajnost
- prenosivost
- efikasnost ...

Postoje 4 generacije programskih jezika:

1. Mašinski jezik
2. Simboličko mašinski jezik(ASEMBLER)
3. Proceduralni programski jezici(algoritamski jezici)
4. Neproceduralni programski jezici

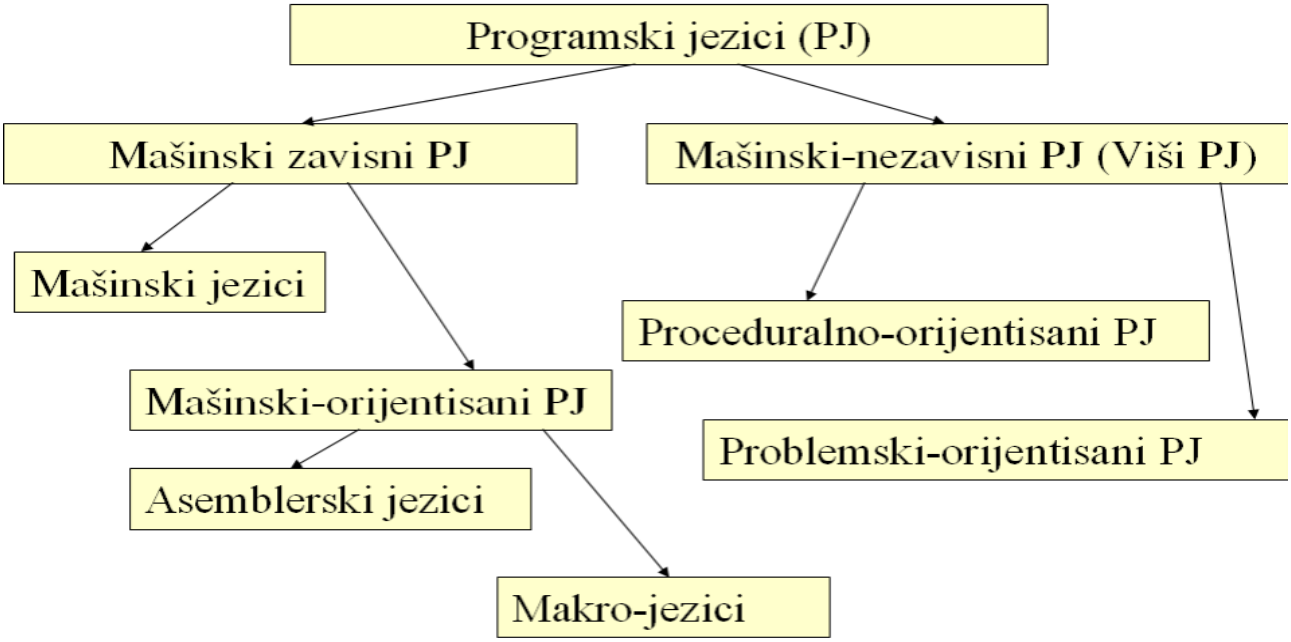
Proceduralni ili viši programski jezici predstavljaju jezike nezavisne od mašine. Način pisanja programa je sličan ljudskom izražavanju.

Dele se na:

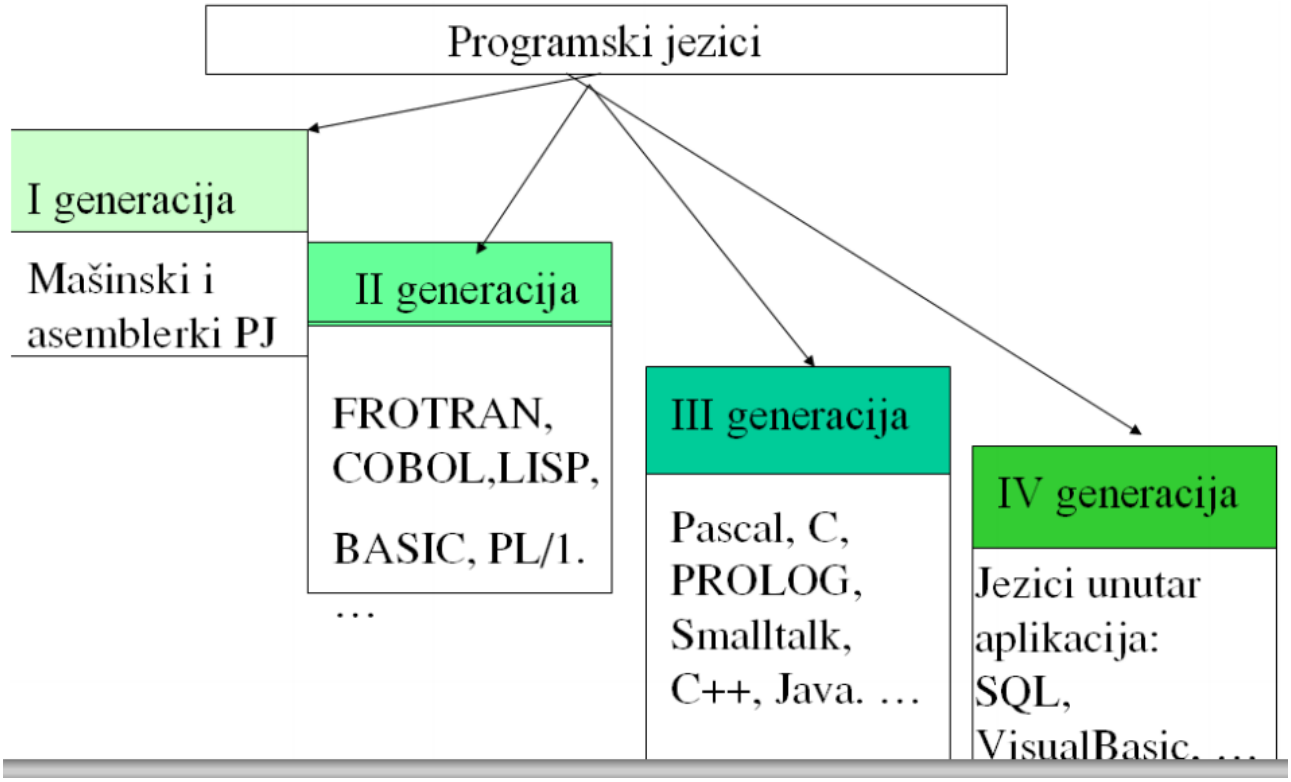
1. Algoritamske programske jezike
2. Problemski orjentisani jezici- za posebne oblasti primene

Do sada je napravljeno nekoliko hiljada programskih jezika. Potrebno je da se svi oni nekako razvrstaju. Kako se računarstvo, pa i programski jezici jako brzo razvijaju, nije lako izvršiti njihovu klasifikaciju. Postoje razne klasifikacije, zavisno od kriterijuma klasifikacije.

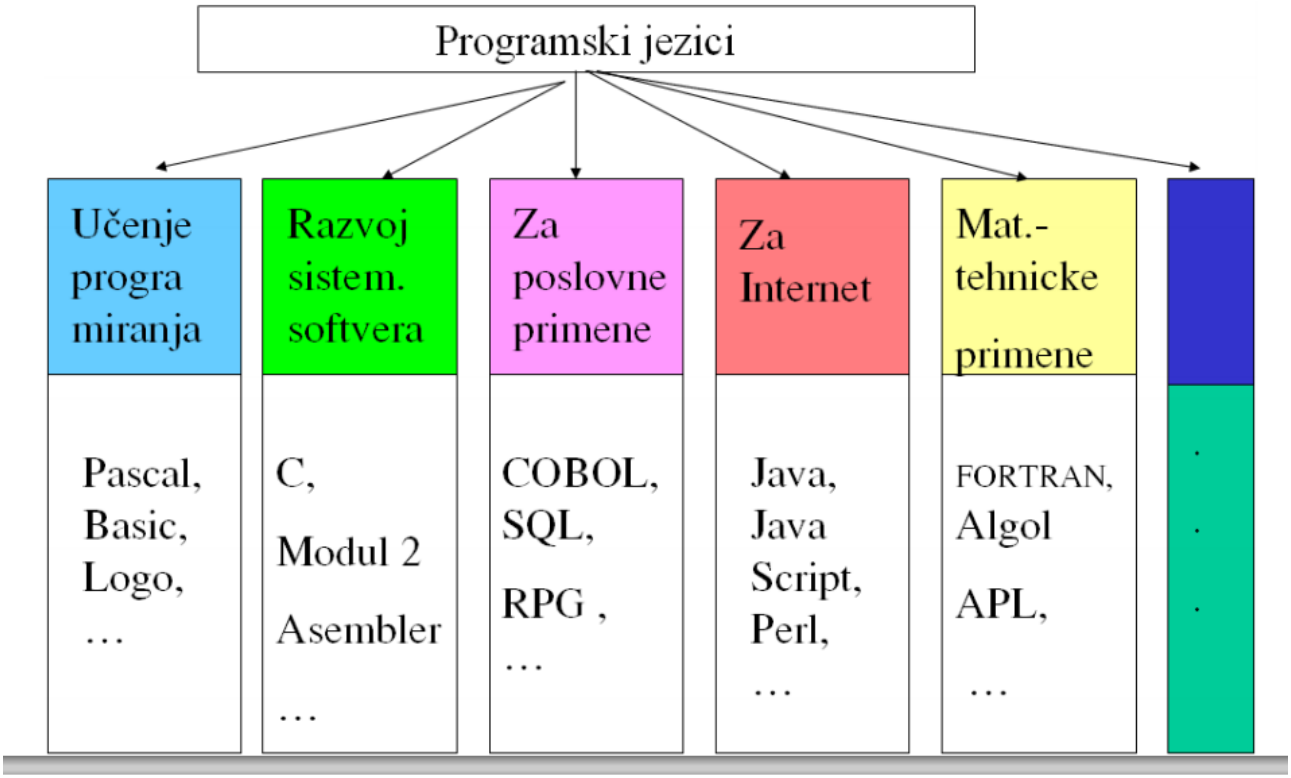
1.Klasifikacija po stepenu zavisnosti od računara



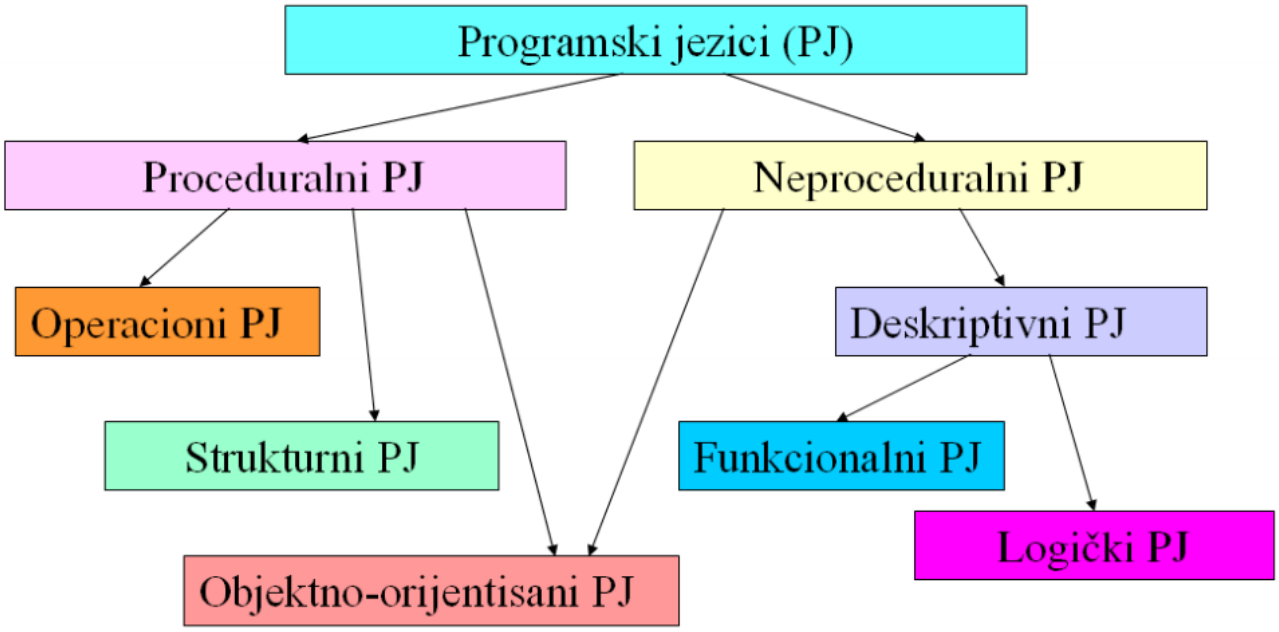
2.Klasifikacija po vremenu nastanka i svojstvima



3.Klasifikacija po oblastima primene

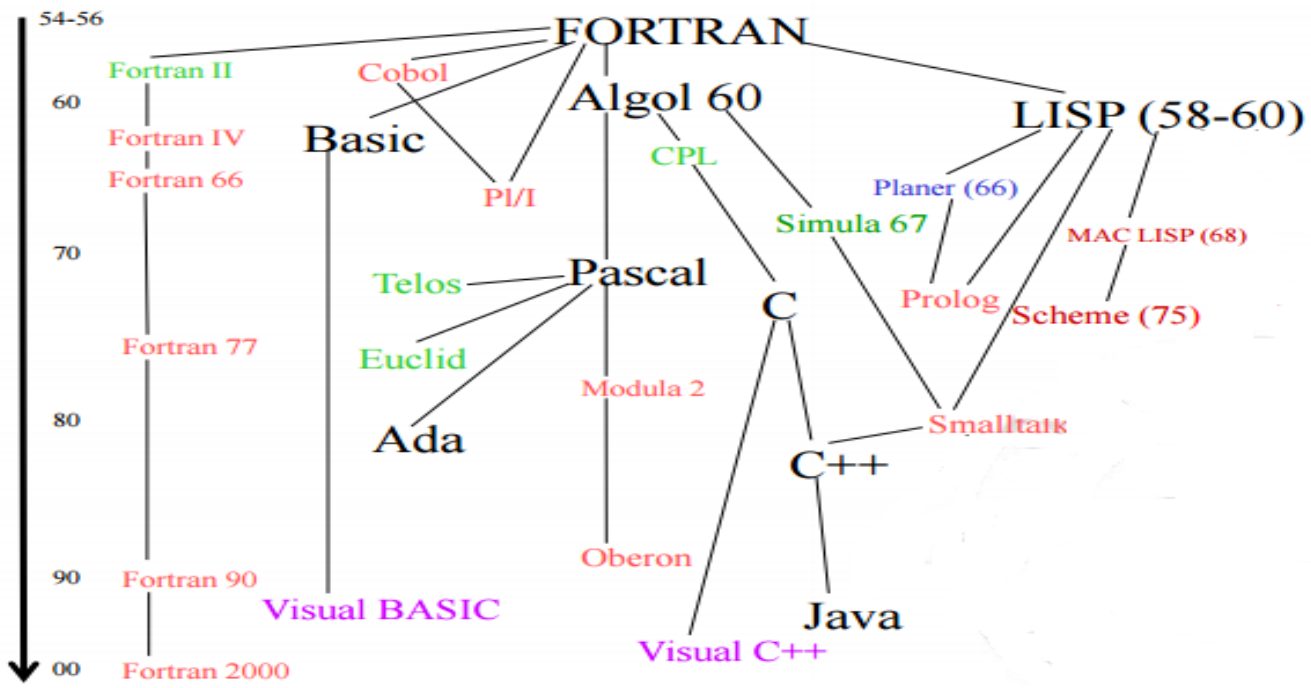


4.Klasifikacija po načinu rešavanja problema



Prikaz razvoja viših programskih jezika

Godina



Računar u toku svog rada prepoznaje i izvršava u procesoru odeređeni broj elementarnih operacije koje se nazivaju **mašinske instrukcije.**

Operacije koje računar izvršava su aritmetičke operacije, logičke operacije ,čuvawe i prenos podataka ,unos,čitawe ,slanje podataka i dr.

Izrazi kojima se zadaju mašinske operacije nazivaju se **mašinske instrukcije ili mašinse naredbe**

Plan delovanja koje trba da obavi neki izvršilac naziva se **PROGRAM.**

Proces pripreme programa, koji se sastoji od projektovanja, pisanja i testiranja programa naziva se **programiranje.**

Faze programiranja:

1. Projektovanje programa(nalaženje niza operacija kojim se rešava konkretan program
2. Pisanje programa –prihvatanje programa na način razumljiv za računar
3. Testiranje programa provera da li program funkcioniše pravilno

Svaki program treba da zadovolji sledeće osnovne osobine:

- da bude što sličniji ljudskom načinu izraćavanja

- da bude formalno definisan ,kako bi se njime mogli jednoznačno izražavati postupci rešenja problema

- da omogući kompaktan i jasan zapis postupka rešenja

- da bude lak za osvajanje

- da omogući formalno prevođenje programa na jezik koji računar rayume – mašinskij jezik

Generacije programskih jezika

Generacija	Opis
prva generacija	mašinski jezik
druga generacija	asemblerski jezik
treća generacija	viši programski jezici (HLL)
četvrta generacija	novi jezici 4GL

Prva generacija – mašinski jezik

Mikroprocesor i drugi logi čki sklopovi ra čunara imaju svoj vlastiti programski jezik koji se naziva mašinski jezik, a sastoji se od nizova binarnih reči koje predstavljaju instrukcije logičkim sklopovima i podatke koje treba obraditi.

Program napisan u mašinskom jeziku nazivamo izvršni program ili izvršni kod jer ga računar može neposredno izvršiti. Mašinski jezik je određen arhitekturom racunara, a daje ga proizvođac hardwarea. Izvršni program je mašinski zavistan, što znači da se kod napisan na jednom racunaru može izvršavati jedino na racunarima istog tipa.

- svaka instrukcija, na hardverskom nivou, direktno upravlja radom mašine, tj. pojedinim gradivnim blokovima.
- instrukcije su numeri čke, predstavljene u formi binarnih oblika od 0 i 1
- programiranje je naporno i podložno velikom broju grešaka
- efikasnost programiranja je niska
- programi nerazumljivi korisniku
- direktno se pristupa resursima mašine
- veća brzina izvršenja programa
- efikasnije korišćenje memorije

Pisanje programa na mašinskom jeziku ima slede će osobine:

• potrebno je izvršiti detaljizaciju algoritma, tako da elementarni koraci odgovaraju pojedinim naredbama, računara,

• uvek je potrebno poznavati skup naredbi datog računara - konkretni mašinski jezik. Program napisan za jedan računar obično se može obavljati na drugom,

• sastavljanje i pisanje programa- kodiranje kao i testiranje programa i podataka u memoriji ra čunara zahteva detaljno poznavanje ra čunara tako da to obavlja odgovaraju ći stručni kadar.

• oblik naredbe je 0001 1101 1000 0000 …

Druga generacija – asemblerski jezik

Karakteristike:

Programiranje na simboli čkom jeziku ima niz nedostataka.

Navedimo najvažnije:

- potrebno je vršiti detaljizaciju algoritma tako da elementarnim algoritamskim koracima odgovaraju dejstva simboli čkih naredbi,
- raznovrsnost ra čunara dovela je do raznovrsnosti simboli čkih jezika, tako da svaki konkretni ra čunar ili familija ra čunara ima svoj poseban simboli čki jezik.

- Program napisan na simboli čkom jeziku za jedan računar ne može bez veće ili manje prerade da se izvršava na drugom.
- svaka instrukcija se predstavlja mnemonikom, kao na primer ADD
- korespondencija izmedju asemblerskih i mašinskih instrukcija je jedan-na-prema-jedan
- postoje i direktive koje nemaju izvršno dejstvo, ali programu na asemblerskom jeziku olakšavaju prevođenje, dodelu memorije i segmentaciju programa
- direktno se pristupa resursima mašine
- veća brzina izvršenja programa
- efikasnije korišćenje memorije

Treća generacija – HLL

Karakteristike:

- kompajler prevodi programske iskaze u odgovarajuće sekvence instrukcija na mašinskom nivou
- u principu jedan iskaz na HLL-u (High Level Language) se prevodi u n (n >= 1) instrukcija na mašinskom (asemblerskom) jeziku o programiranje je jednostavnije
- efikasnost je veća o
- ispravljanje grešaka lakše
- nema direktni pristup resursima mašine
- neefikasno iskorišćenje memorije o
- duži programi
- Programiranje je znatno olakšano.
- Ovi jezici su bliski čovekovom pisanom jeziku i operativnoj terminologiji iz odgovarajuće oblasti,skraćeno je vreme obuke u programiranju i izbegnute su teškoće oko detalja u vezi sa programiranjem na mašinskom, odnosno simboličkom jeziku,
- Ovi jezici su nezavisni od strukture samog računara.
- Program napisan na ovom jeziku može da se izvršava na svakom računaru koji ima prevodilac (kompajler) za ovaj jezik,
- Postoji mogućnost izmene programa i iskustva između korisnika jednog problemsko - orjentisanog jezika.

Četvrta generacija – novi jezici 4GL

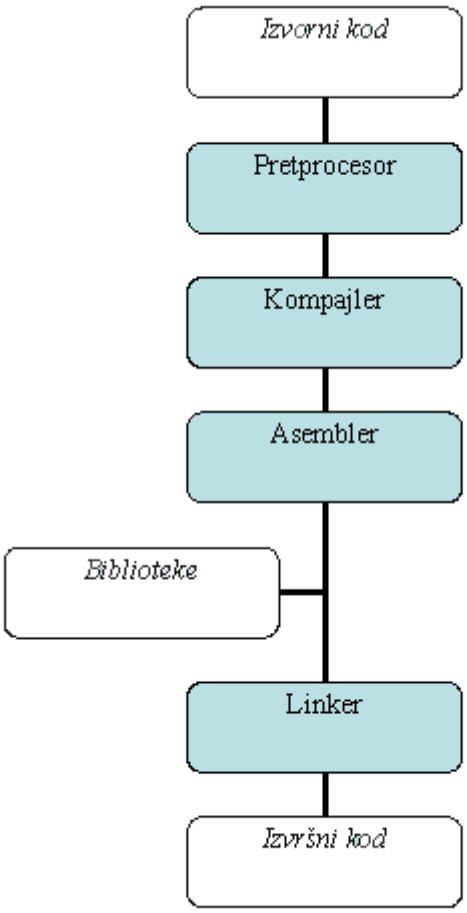
Novi tipovi računarskih jezika se karakterišu sledećim osobinama:

- implementiraju veštačku inteligenciju (primer je LISP) o jezici za pristup bazama podataka (primer je SQL)
- objektno-orijentisani jezici (primeri su C++, Java i dr.)

Podela programskih jezika prema oblasti primene

- naučne aplikacije,
- poslovnu obradu,
- veštačku inteligenciju,
- projektovanje sistemskog softvera,
- projektovanje pomocu računara (CAD),
- upravljahje pomocu računara (CAM),
- opis hardvera računara,
- simboličko programiranje,
- linearno programiranje,
- simulaciju,
- računarske komunikacije,
- prostorno planiranje,
- stono izdavaštvo,
- obradu teksta (tekst procesori)

PROCES PREVODJENJA PROGRAMA – C JEZIK



Pretprocesor

Učićemo o ovom delu procesa kompilacije detaljnije kasnije. Međutim, sada nam trebaju osnovne informacije za neke programe u C-u.

Pretprocesor čita Vaš izvorni kod i odgovoran je za uklanjanje komentara i interpretaciju specijalnih pretprocesorskih direktiva koje se označavaju sa #.

Na primer

- #include – uključuje sadržaj imenovanog fajla. Npr. #include <math.h> uključuje standardnu biblioteku sa matematičkim funkcijama, #include <stdio.h> uključuje standardnu biblioteku za upravljanje ulazom i izlazom
- #define – definiše simboličko ime ili konstantu. Npr. #define MAX 100 definiše konstantu MAX=100

Kompajler

Kompajler prevodi izvorni kod u asemblerski. Izvorni kod dobija od pretprocesora.

Asembler

Asembler pravi objektni kod. To je fajl sa ekstenzijom .obj.

Linker

Ako se u izvornom kodu pozivaju bibliotečke funkcije ili funkcije definisane u nekom drugom izvornom fajlu linker kombinuje ove funkcije (sa funkcijom main()) da bi napravio izvršni program. Pozivanje spoljnih promenljivih se takođe rešava ovde. Više o tome kasnije.

Korišćenje biblioteka

C je ekstremno mali jezik. Mnoge funkcije drugih jezika nisu uključene u C. Npr. nema funkcija za učitavanje podataka sa tastature, ispis na ekran, rad sa stringovima, ili matematičkih funkcija.

Čemu onda služi C?

C omogućava rad kroz bogatu zbirku biblioteka sa funkcijama.

Rezultat toga je što mnogi kompajleri uključuju standardne biblioteke funkcija za mnoge namene (Ulaz/izaz, …). Iz mnogih praktičnih razloga o njima razmišljamo kao da su deo C-a. Međutim one se mogu razlikovati od mašine do mašine.

Programer može takođe razviti svoje biblioteke sa funkcijama i uključiti neke druge specijalne biblioteke drugih ljudi.

Pronalaženje informacija o bibliotečkim funkcijama

U help sekciji editora možete pretraživati po funkcijama koje su Vam potrebne da biste našli u kojoj biblioteci se nalaze.

Ako ne znate ime funkcije koja Vam je potrebna, možete naći spisak svih biblioteka pod „header files“. Tu ima oko 700 opisanih funkcija. Ovaj broj ima tendenciju rasta sa svakom nadogradnjom kompajlera.