

UNIVERZITET U ZENICI

Katedra za matematiku i informatiku

Senad Balić
Denis Spahić

PRINCIPI PROGRAMIRANJA

Liberty BASIC

Odabrana poglavlja

Zenica, april 2006. god.

1. UVOD U PROGRAMIRANJE

1.1 Uvodne napomene

Učiti kako programirati, a poslije to i samostalno raditi, lakše je nego što se obično misli.

Potrebno je ovladati određenim programskim vještinama i određenim principima u programiranju, odnosno u kreiranju programa, na što je moguće jednostavniji način.

Cilj kursa

Edukacija studenata iz područja principa programiranja i njihovo ovladavanje programskim vještinama i principima kreiranja programa, s ciljem rješavanja problema i zadataka uz podršku računara.

Cilj kursa	Edukacija studenata iz područja principa programiranja i njihovo ovladavanje programskim vještinama i principima kreiranja programa, s ciljem rješavanja problema i zadataka uz podršku računara.
Zadaci tokom studija	Student treba: <ul style="list-style-type: none">- da nauči nekoliko programskih jezika koji se izučavaju i primjenjuju u odgojno-obrazovnom sistemu,- da se obuči za praktičnu primjenu pomenutih programskih jezika, kao sredstva za vlastiti rad i za obuku u obrazovnom procesu,- da se osposobi za kreiranje aplikacija uz pomoć pomenutih programskih jezika, i- za rješavanje problema za potrebe nastavnog procesa svih predmeta nastavnog programa.
Sadržaj kursa	<ul style="list-style-type: none">- Programiranje i programski jezici,- Tehnike, metodika i stil programiranja,- Rješavanje problema uz podršku računara,- Programski jezik Liberty BASIC,- Kreiranje ili dizajn programa,- Algoritamske strukture,- Uvod u numeričke metode i programiranje,- Veza s programskim jezikom Visual BASIC,- Vizuelno programiranje,- Budućnost programiranja

1.2 Programiranje i programski jezici

1.2.1 Pojam programa, pojam instrukcije, način izvršenja programa

Program je niz instrukcija napisanih određenim redoslijedom, tako da kao cjelina izvršavaju neki zadatak pretvaranja ulaznih u izlazne podatke.

Programiranje u užem smislu riječi predstavlja pisanje programa nekim programskim jezikom.

Programski jezik je set instrukcija koje računar razumije i koje može da interpretira.

Instrukcije se dijele na:

- aritmetičke (sa fiksnim i pokretnim zarezom),
- logičke (poredbene),
- organizacijske (ulazno/izlazne i transportne) i
- kontrolne (potprogrami, IF-THEN).

1.2.2 Generacije programskih jezika, vrste prevodilaca

Programski jezici se dijele na niže i na više programske jezike.

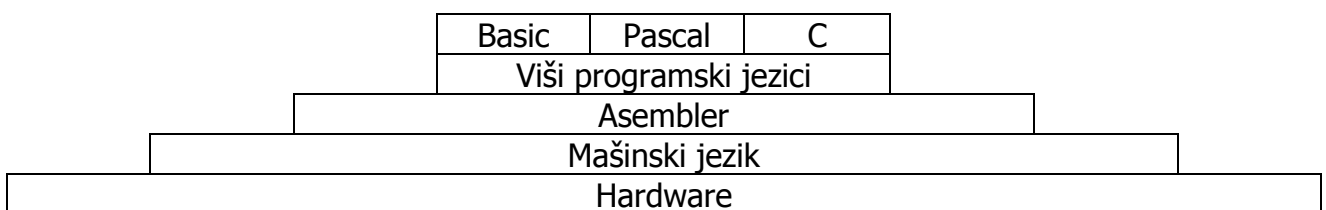
Niži programski jezik je mašinski jezik,

dok u više programske jezike spadaju: BASIC, FORTRAN, PASCAL, COBOL, C, C++, itd.

Niži programski jezici su platformski, okrenuti računaru (instrukcije se opisuju simbolički), dok su viši programski jezici problemski i bliži su korisniku (instrukcije su obično izvedene iz riječi engleskog jezika).

Generacije programskih jezika:

- mašinski jezik se naziva i programski jezik prve generacije (1GL, 1950-1954),
- assembler i makro-assembler su jezici druge generacije (2GL, 1955-1959) i nalaze se između nižih i viših jezika,
- jezici treće generacije (3GL) su pomenuti viši programski jezici, koji su proceduralni,
- u jezike četvrte generacije (4GL) spadaju: SQL, HTML, PHP, ASP, tj. neproceduralni jezici sa usko specijaliziranom namjenom:
 - opisni (služe opisivanju dokumenata - PostScript, HTML),
 - upitni (generisanje podskupova iz baza podataka - SQL),
 - grafički (LabView, G – jezici za programiranje virtualnih instrumenata).



Računar može da izvršava samo programe napisane mašinskim jezikom (set instrukcija koje razumije mikroprocesor), tako da je sve programe napisane nekim od viših programskih jezika potrebno prevesti u mašinski jezik.

Izvorni kod predstavlja niz instrukcija napisanih višim programskim jezikom, dok izvršni kod predstavlja datoteku koja sadrži instrukcije mašinskog jezika.

Prema vremenu prevođenja instrukcija u mašinski jezik, prevodioci se dijele na:

- kompajlere (compiler) i
- interpretere.

Interpreter prevodi instrukciju neposredno nakon njenog izdavanja, tj. u RAM memoriji računara se nalazi izvorni kod programa, koji se mora prevoditi svaki put kad se program pokrene.

Kompajler prevodi kompletan izvorni kod i snima ga u datoteku, tako da se u RAM memoriji računara ne nalazi izvorni kod nego samo mašinske instrukcije.

1.2.3 Programski jezik BASIC

Programski jezik BASIC (**B**eginners **A**ll-purpose **S**ymbolic **I**nstruction **C**ode) je razvijen u SAD oko 1970. godine, a postoji veliki broj verzija, različitih proizvođača (Borland, Microsoft,...).

Prvi kućni računari su imali BASIC u ROM memoriji, dok još nije bilo operativnih sistema (za trajno pohranjivanje podataka nisu se koristili diskovi, nego audio kasete i magnetne trake).

Razlike između raznih verzija BASICa su vrlo male, a svode se na različitu sintaksu pojedinih naredbi, te postojanje nekih instrukcija koje ne postoje u drugim verzijama.

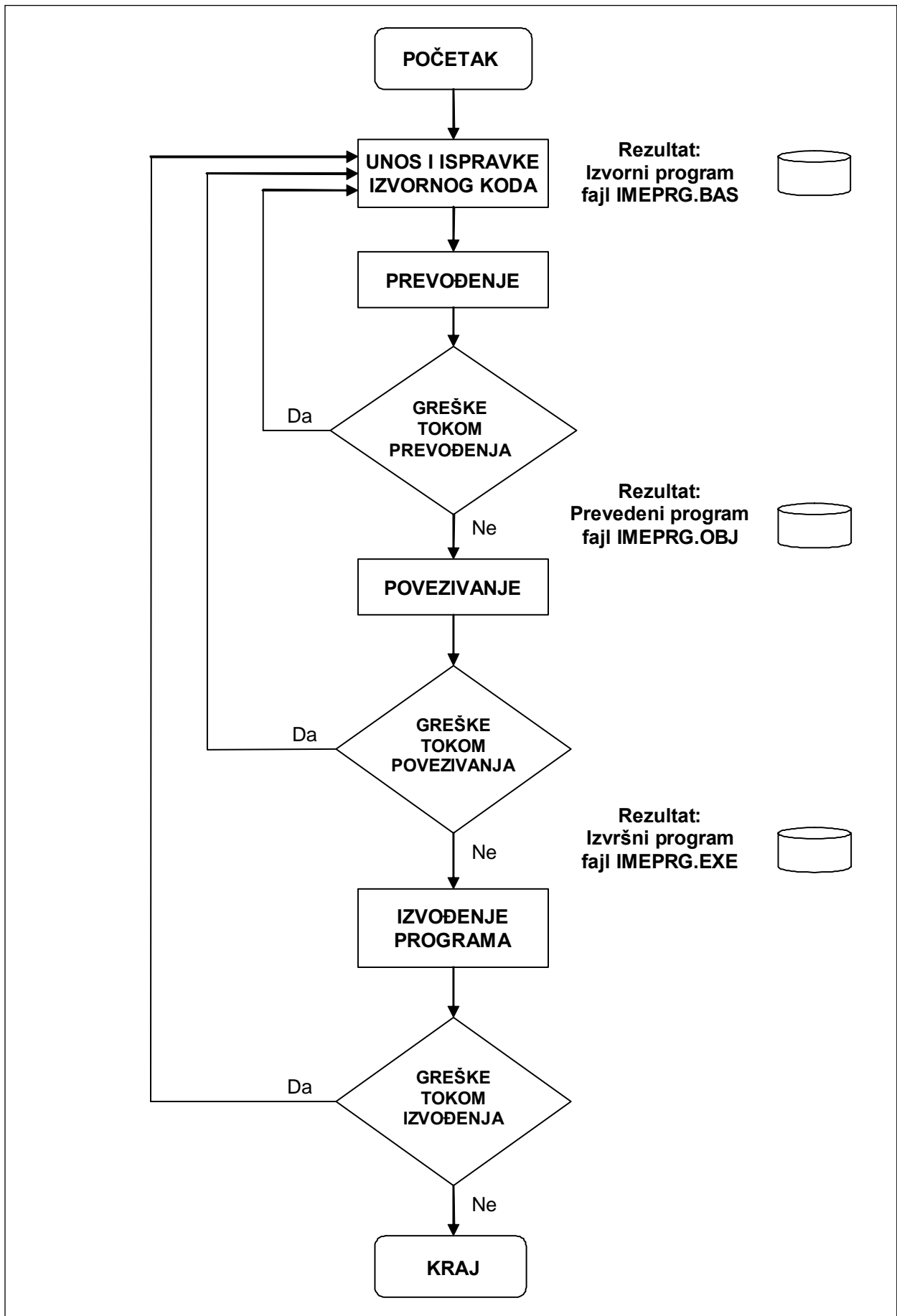
Starije varijante BASIC-a su se iz praktičnih razloga koristile brojevima za označavanje naredbi (na početku naredbi), jer nije bilo editora za uređivanje teksta programa. Naredbe su se numerisale brojevima 10, 20, 30,... Ako bi trebalo dodati naredbu između 20 i 30, toj novoj naredbi dao bi se broj 25, i ona bi automatski bila smještena u memoriju na mjesto koje joj pripada po redoslijedu brojeva naredbi.

U Liberty BASICu ne mora se koristiti numerisanje naredbi, jer se program uređuje pomoću solidnog editora.

1.2.4 Postupak kreiranja izvršnog programa i njegovo testiranje

Postupak kreiranja izvršnog programa, kod kojeg se polazi od izvornog programa (od izvornog koda se dobiva izvršni, odnosno binarni kod) se sastoji iz slijedećih faza:

- pisanje izvornog programa, odnosno izvornog koda u editoru,
- prevođenje ili kompajliranje (compiling) izvornog programa - koda, kojim se dobija objektni kod. Tokom kompajliranja provjerava se i sintaksa instrukcija, tj. da li su sva imena instrukcija pravilno napisana i da li su svi argumenti uneseni na dozvoljeni način;
- povezivanje (linking) objektnog koda sa bibliotekama gotovih funkcija (nalaze se u instaliranoj aplikaciji prevodioca), koje je potrebno da bi računar mogao izvršiti funkcije i određene instrukcije (na primjer funkcije: SIN, COS, LOG ili ulazno/izlazne instrukcije) i
- testiranje programa, kada se ispituje da li je program moguće izvršiti: da li će se javiti dijeljenje s nulom, korjenovanje negativnog broja, ili slični slučajevi nerješivi za računar.



Sl. 1.1 Tok razvoja programa