Osnove inženjerske informatike II.

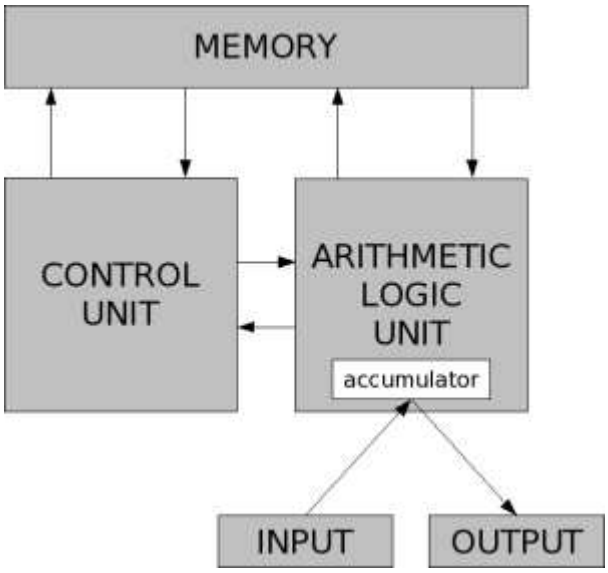
Uvod u programiranje

1. predavanje:

Programski jezici

K. F.

Von Neumannova arhitektura računala (engl. *Von Neumann architecture*):



1. glavne jedinice računala su upravljačka i aritmetičko–logička jedinica (koje zajedno čine procesor računala), te memorija i ulazno–izlazni uređaji;
2. program i podaci nalaze se u istoj memoriji kodirani na isti način;
3. program se sastoji od niza naredbi koje se izvode nad podacima; u nekom trenutku izvodi se samo jedna naredba, a redoslijed izvođenja naredbi određuje upravljačka jedinica (neke suvremene višeprocesorske arhitekture su tek sklopovi međusobno usklađenih von Neumannovih strojeva)

podatak (engl. *datum*, češće u množini *data*): opći naziv za vrijednosti i druge jedinice značenja, sastavljene od brojeva, slova i drugih simbola, prilagođene obradi računalom (pohranjivanju, prikazu, prijenosu i preoblikovanju)

kompjutorski program (engl. *computer programme*, češće amer. engl. *computer program*):

- uputa prema kojoj računalo može izvesti određeni zadatak ili riješiti određenu zadaću

izvedbeni ili **izvršni program** (engl. *executable program*):

- cjeloviti niz instrukcija *strojnog jezika* koje računalo može neposredno izvesti

izvorni program (engl. *source program*):

- program napisan u *mnemoničkom* ili u *višem programskom jeziku*, koji računalo može izvesti tek nakon prevođenja u izvršni program;
- definicija sadržaja izvornog programa u višem jeziku ovisi o *programskoj paradigmi* koju jezik podržava

programska paradigma (engl. *programming paradigm*):

- skup teorijskih pretpostavki o tome je zapravo „izračunavanje" kao i preporuka i „zabrana" te obrazaca, primjera i mjerila za sustavno oblikovanje i pisanje programa
- osnovna (iako djelomična) klasifikacija:
  1. proceduralne paradigme — proceduralno programiranje,
  2. izjavne paradigme — izjavno programiranje,
  3. objektne i srodne paradigme

programski jezik (engl. *programming language*):

- skup znakova, riječi i pravila namijenjen nedvosmislenom zapisivanju kompjutorskih programa;
- umjetni jezik strogo određene sintakse i semantike;
- **sintaksa** (engl. *syntax*) — skup pravila koja propisuju poredak, međusobni razmještaj i povezivanje simbola u izraze;

  <datum> := `ZZ/ZZ/ZZZZ`

  <Z> := `0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9`

  `01/02/2006` u redu

  `010/A/20061` nedopušteno
- **semantika** (engl. *semantics*) — skup pravila koja definiraju značenja simbola i sintaktički pravilno oblikovanih izraza;

  `01/02/2006` $\Longrightarrow$ 1. veljače 2006. kod nas

  `01/02/2006` $\Longrightarrow$ 2. siječnja 2006. u Americi

  `27/03/2006` $\Longrightarrow$ danas kod nas

  `27/03/2006` $\Longrightarrow$??? u Americi

- razine jezikâ:
  1. strojni jezik (engl. *machine language*) — skup instrukcija u binarnom kôdu,
  2. mnemonički jezik (engl. *assembly language*) — strojne instrukcije se označavaju kraticama njihovih engleskih naziva,
  3. viši programski jezici — sintaksa i simbolika prilagođene su područjima primjene, npr. FORTRAN (FORmula TRANslator) za numeričke programe;

- klasifikacija obično prema paradigmama koje podržavaju (jezik *podržava* neku paradigmu ako se u njemu mogu neposredno, pregledno i čitljivo pisati programi koji se pridržavaju „zakona" paradigme te ako pružaju zaštitu od odstupanja od tih zakona)

  „. . . pripadnik nekog društva — koji, naravno, kodira proživljenu stvarnost uporabom određenoga jezika i drugih simboličkih oblika svojstvenih njegovoj kulturi — može shvatiti stvarnost samo u obliku u kojem se prikazuje u tom kôdu. To ne znači da je stvarnost sama po sebi relativna, nego da joj pripadnici različitih kultura daju različite naglaske i da je različito kategoriziraju ili da zamjećuju različite vidove stvarnosti."

  Dorothy Lee

proceduralni program (engl. *procedural program*):

- niz naredbi programskoga jezika čiji unaprijed utvrđeni *slijed* opisuje *uzastopne korake transformacije* ulaznih podataka u tražene rezultate;
- formalni zapis *algoritma*
- jezici: FORTRAN 77, ALGOL, C, Pascal; C++, Java

algoritam (engl. *algorithm*):

- razgovijetan, potpun i nedvosmislen opis postupka rješavanja određene zadaće u obliku niza jasno definiranih koraka
- osnovna svojstva/zahtjevi:
  1. *konačnost* — izvođenje algoritma, neovisno o konkretnom skupu ulaznih/početnih podataka, mora završiti u konačnom broju koraka;

2. *jedinstveni početak* — svako izvođenje algoritma mora početi istim korakom;
3. *jedinstveni nastavak* — iza svakog koraka mora slijediti jednoznačno određeni korak (tj. niz koraka ne može biti slučajan);
4. *rješenje* — svako izvođenje algoritma mora dovesti do rješenja ili završiti s naznakom da je za zadani skup početnih podataka zadaća nerješiva tim algoritmom

proceduralni program:

- primjer: zbrajanje vektorâ

```
double da[n], db[n], dc[n];
// initialize da & db
for (int i = 0; i < n; i = i + 1)
   dc[i] = da[i] + db[i];

dcomplex ca[n], cb[n], cc[n];
// initialize ca & cb
for (int i = 0; i < n; i = i + 1)
   cc[i] = ca[i] + cb[i];
```

objektne i srodne paradigme: uvođenje novih tipova podataka (razreda) koji odgovaraju pojmovima područja primjene pa se njihovi predstavnici (objekti) „znaju ponašati" u skladu sa svojim ulogama;

1. objektno utemeljeno programiranje (engl. *object based programming*)
   — apstrakcija podataka: uvođenje novih tipova podataka,
   — jezici: Simula 67, C++, Ada, Fortran 90;
2. objektno usmjereno programiranje (engl. *object oriented programming*)
   — hijerarhija razreda / nasljeđivanje,
   — jezici: Simula 67, C++, Java, Smalltalk, Python;
3. programiranje utemeljeno na vrstama/rodovima (engl. *generic programming*)
   — skup zahtjeva koje neki izraz mora zadovoljiti da bi se mogao pojaviti u nekom drugom izrazu,
   — jezici: C++, Ada, (Fortran 90)

apstrakcija podataka:

- primjer: vektor je uveden kao novi tip podataka

```
class dvector;
class cvector;

dvector operator+ (dvector const&, dvector const&);
cvector operator+ (cvector const&, cvector const&);

dvector da(n), db(n), dc(n);
// initialize da & db
dc = da + db;

cvector ca(n), cb(n), cc(n);
// initialize ca & cb
cc = ca + cb;
```

vrste:

- primjer 1.: tip kojem pripadaju elementi vektora mora imati definirano zbrajanje

```
template<typename T>
class vector;

template<typename T>
vector<T> operator+
(vector<T> const&, vector<T> const&);

vector<double> da(n), db(n), dc(n);
// initialize da & db
dc = da + db;

vector<complex<double> >  ca(n), cb(n), cc(n);
// initialize ca & cb
cc = ca + cb;
```

- primjer 2.: izraz `f` je *unarna funkcija* ako se može pojaviti u funkcijskom pozivu `f(x)`:

  `sin` u `sin(x)`

  `g`, ali i `h` u `g(h(x))`

  `compose(g,h)` u `compose(g,h)(x)`

  `compose` je *binarna funkcija* čiji su parametri dvije unarne funkcije, a vrijednost nova unarna funkcija

izjavni program (engl. *declarative program*):

- opis *odnosa* ulaznih podataka i traženih rezultata ili *uvjeta* koje rezultati moraju zadovoljavati;
- podjela:
  1. funkcijski programi,
  2. logički programi

funkcijski program (engl. *functional program*):

- (neuređeni) skup definicija funkcija i vrijednosti;
- funkcije se definiraju pomoću drugih funkcija i vrijednosti;
- funkcije mogu biti argumenti i/ili rezultati drugih funkcija, što omogućuje simbolička, a ne samo numerička izračunavanja (primjerice, simboličko deriviranje i integriranje, gdje je rezultat funkcija, a ne numerička vrijednost);
- jezici: Lisp, Scheme, ML, Miranda, Haskell; (C++)

logički program (engl. *logic program*):

- (neuređeni) skup relacija među vrijednostima i simbolima izraženih činjenicama i pravilima u obliku

  zaključak ako *pretpostavke*

- jezici: Prolog, Gödel

program prevoditelj (engl. *compiler*):

- program koji program napisan u višem programskom jeziku prevodi u strojni jezik;
- rezultat prevođenja najčešće nije izvršni program nego datoteka s tzv. ciljnim kôdom (engl. *object code*) koji sadrži mješavinu instrukcija u strojnom jeziku i simboličkih informacija s pomoću kojih u sljedećem koraku povezivač (engl. *linker*) ciljnoj datoteci dodaje modul koji omogućava pokretanje programa, te pretražuje biblioteku standardnih funkcija i u izvršni program povezuje module s potrebnim funkcijama

program tumač (engl. *interpreter*):

- program koji program napisan u višem programskom jeziku u strojni jezik prevodi naredbu po naredbu i odmah je izvodi