6. OSNOVE PROGRAMIRANJA

6.1. Uvod

Za pravilan i učinkovit rad računala potrebno je:

* strojna oprema (hardware),

zajedno čine računalni sustav

* programska oprema (software)

**Programska oprema** računala je skup programa koji se mogu izvršiti na računalu. Svaki od njih obavlja određenu funkciju.

**Program** je niz naredbi koje se izvode točno određenim redoslijedom i s točno određenim ciljem. **Naredba** je temeljni element programa. Naredbama se točno opisuje *što* računalo treba raditi i *kako* to izvesti.

Putem programa korisnik upravlja računalom ako ih zna pravilno koristiti.

Programe pišu stručnjaci koji se zovu **programeri**. Pri tome se koriste nekim od **programskih jezika** koje računalo razumije i programskim naredbama koje računalo može izvršiti.

Postupak stvaranja programa naziva se **programiranje**.

6.2. Programski jezici

Dakle, računala se također sporazumijevaju jezicima.

Jezici koje računala razumiju zovu se **programski jezici**.

**Programski jezik** je skup ključnih riječi i pravila za njihovo korištenje koje “razumije” računalo.

Postoji nekoliko **generacija programskih jezika**:

6.2.1. Prva generacija: strojni jezici

Pisanje programa za prva računala bilo je isključivo u strojnom jeziku.

**Strojni jezik** je binarni prikaz programa za računalo, dakle, jezik nula i jedinica

To je najniža moguća razina prikaza programa za neko računalo, pritom je usko vezana uz građu računala.

Pisanje programa u strojnom jeziku bilo je otežano nerazumljivim kodovima pojedinih naredbi (svaka naredba je kombinacija nula i jedinica). Zbog toga je i vjerojatnost pogreške pri pisanju programa bila velika. Od programera je zahtijevalo dobro poznavanje građe računala. A nije postojao ni jedinstven strojni jezik, već je svaki stroj imao sebi svojstven strojni jezik..

6.2.2. Druga generacija: simbolički (asemblerski) jezici

Prvi korak ka “jezicima više razine” bilo je uvođenje **simboličkog (asemblerskog) jezika.**

U tom jeziku programer koristi riječi za operacije i nepoznanice za podatke.

Npr. ako treba zbrojiti dva broja, a mjesta u memoriji na kojima su ti brojevi zapisani označimo s X i Y, tada u asemblerskom jeziku pišemo *ADD X,Y* umjesto *0110 001110 010101* u nekom strojnom jeziku.

Razumijevanje programa time postaje puno lakše jer su numerički kodovi sastavljeni od nula i jedinica zamijenjeni riječima koje podsjećaju na operaciju koju treba napraviti (npr. ADD) i simbolima koji označavaju nepoznanice.

6.2.3. Treća generacija: viši programski jezici

**Viši programski jezici** omogućavaju programeru da piše programe u prirodnijem zapisu jer se skup naredbi strojnog, odnosno asemblerskog jezika zamjenjuje jednom naredbom višega programskog jezika. Pritom se od programera ne zahtijeva poznavanje građe računala, a isti program se može primijeniti na različitim modelima računala.

Takvi jezici nisu namijenjeni strogo uskom području primjene već je njima moguće rješavati relativno širok raspon zadataka.

Neki od jezika koji se ovdje ubrajaju su: BASIC, Pascal, C (C++), LOGO, Fortran, Ada, LISP, PROLOG,...

Npr.

želimo li da računalo zbroji dva broja i rezultat zapamti na nekom trećem mjestu u memoriji, pisat ćemo naredbu tipa

*C=A+B*

što znači: zbroji brojeve koji su zapisani na A i B, a rezultat stavi na C.

Pogledajmo kako u cijelosti izgleda program za zbrajanje dva cijela broja u nekim od viših programskih jezika:

|  |  |
| --- | --- |
| **QBASIC:** | **LOGO:** |
| INPUT A  INPUT B  C=A+B  PRINT C  END | TO ZBROJI  MAKE “A READ  MAKE “B READ  MAKE “C :A+:B  PR :C  END |
| **PASCAL:** | **C++:** |
| program zbroji;  var a, b: integer;  begin  readln (a);  readln (b);  c:=a+b;  writeln (c);  end. | #include <stdio.h>  main ()  {  int a,b,c;  scanf ("%d,%d", &a, &b);  c=a+b;  printf ("%d", c);  } |

Budući da računalo razumije samo strojni jezik, da bi moglo izvršiti program napisan u bilo kojem od ovih jezika, potreban je "prevodilac" koji bi naš program preveo u strojni jezik. Ti "prevodioci" su isto programi koji se još zovu **jezični prevoditelji**. Postoje dvije skupine takvih jezika za prevođenje, a to su:

* interpreteri,
* **kompajleri**.

6.2.4. Četvrta generacija: jezici prilagođeni krajnjim korisnicima

Ovi programski jezici potpuno su prilagođeni krajnjim korisnicima koji u principu nisu programeri, a uspješno se njime služe i programeri radi ubrzanja procesa programiranja.

Primjeri takvih jezika su:

* upitni jezici za bazu podataka (SQL, Asi, GIS,…)
* jezici za podršku u odlučivanju (visiCalc, Multiplan, …)
* generatori programa (Mapper, Ramis II, Focus, …)
* jezici za programiranje vrlo visoke razine (AME, NOMAD, …)
* jezici za crtanje

6.3. Algoritam

Svaki problem koji želimo riješiti uz pomoć računala treba prvo raščlaniti na dijelove ili operacije koje računalo razumije. Postupak kojim računalo rješava neki problem zove se **algoritam**.

Algoritam je stara riječ koja potječe iz arapskog jezika, a znači *postupak, pravilo, uputa.*

Različite algoritme susrećemo i u svakodnevnom životu.

Pogledajmo primjer algoritma za pečenje kolača:

POČETAK

POTRAŽI maminu kuharicu

PROVJERI imaš li kod kuće sve potrebne sastojke:

jaja, šećer, brašno, maslac

AKO nemaš sve sastojke,

ONDA odi u trgovinu

INAČE počini peći kolač

UZMI zdjelu, kuhaču i vagu

ODIJELI žumanjke od bjelanjaka

DODAJ šećer

DODAJ maslac

sve dobro izmješaj

ISTUCI snijeg od bjelanjaka

DODAJ po jednu žlicu

snijega

brašna

DOK ima sastojaka

UZMI posudu za pečenje

STAVI smjesu u posudu

UKLJUČI pećnicu

DOK tijesto nije pečeno

peci na laganoj vatri

IZVADI kolač

POŠEĆERI kolač

KRAJ

Dakle, **algoritam** je slijed radnji ili postupaka koji dovode do rješenja postavljenog zadatka.

Uz svaki algoritam moraju jasno biti definirani **početni objekti** (ulazni podaci) nad kojima se obavljaju operacije (iz primjera: jaja, šećer, brašno, maslac, kuhača, posuda, kuharica). Kao ishod provođenja algoritma pojavljuju se **završni objekti** ili rezultati (iz primjera: kolač).

Algoritam mora biti sastavljen od *konačnog broja koraka* koji ukazuju na slijed operacija koje treba obaviti nad početnim objektima kako bi se dobili rezultati. Svaki korak opisuje se **instrukcijom** (iz primjera: POTRAŽI maminu kuharicu, DODAJ šećer,...).

Obavljanje algoritma naziva se **algoritamski proces**. Tijekom odvijanja algoritamskog procesa mogu se pojaviti i neki **međurezultati** (iz primjera: ljuske od jaja, prljava zdjela).

Za obavljanje algoritma potreban je **izvoditelj algoritma** koji *razumije algoritam* i znade *točno obaviti svaki korak algoritma* (iz primjera: domaćica, kuharica).

Algoritmi mogu biti:

* **specijalizirani** – mogu se primijeniti samo na pojedine početne objekte (primjer pečenja kolača);
* **općeniti** – dozvoljavaju različite vrijednosti početnih objekata s tim da se kod takvih algoritama definira **klasa ulaznih objekata** koji su dozvoljeni (primjer zbrajanja dva broja iz poglavlja 6.2.3., klasa ulaznih objekata su cijeli brojevi).

**Zadatak:** Napiši algoritam za:

* 1. prelaženje ulice
  2. jutarnje buđenje
  3. sjetvu povrtnih kultura
  4. kuharski recepti (kuhanje kave i sl.).
  5. traženje dečka ili djevojke
  6. slanje SMS poruke

Zaključak: može se reći da je program zapravo algoritam zapisan u nekom od programskih jezika.

Evo i jedne tablice za usporedbu:

|  |  |
| --- | --- |
| ALGORITAM | PROGRAM |
| početni objekti | ulazni podaci |
| završni objekti | izlazni podaci |
| klasa objekata | tip podataka |
| instrukcije | naredbe |

6.4. Osnovni algoritamski postupci

Postoje tri osnovna algoritamska postupka:

* slijed
* grananje
* ponavljanje (petlja)

6.4.1. Slijed

Naredbe slijede jedna iza druge.

Iz primjera "pečenje kolača":

UZMI zdjelu, kuhaču i vagu

ODIJELI žumanjke od bjelanjaka

DODAJ šećer

DODAJ maslac

6.4.2. Grananje

Kod grananja se prvo postavlja neki uvjet, a zatim se ovisno o rezultatu uvjeta izvode sljedeće naredbe.

Uvjet je uvijek takav da se na njega može odgovoriti sa DA ili NE. Takve uvjete zovemo **logički uvjeti**.

Iz primjera "pečenje kolača":

AKO nemaš sve sastojke,

ONDA odi u trgovinu

INAČE počni peći kolač

6.4.3. Ponavljanje

Kod ponavljanja određeni niz naredbi izvršava se zadani broj puta.

Pri tome možemo imati dvije situacije:

* prije početka ponavljanja unaprijed se točno zna koliko puta postupak treba ponoviti,
* broj ponavljanja ovisi o izvršavanju niza naredbi koje se ponavljaju i tada se broj ponavljanja ne zna unaprijed.

Niz naredbi koje se ponavljaju zajedno s naredbama koje određuju ponavljanje najčešće zovemo **petlja**.

Iz primjera "pečenje kolača":

DOK tijesto nije pečeno

peci na laganoj vatri

DODAJ po jednu žlicu

ili

snijega

brašna

DOK ima sastojaka

6.4.4. Podjela instrukcija

Program se sastoji od niza naredbi, a algoritam od niza instrukcija.

Instrukcije se mogu podijeliti u nekoliko skupina.

|  |  |  |  |
| --- | --- | --- | --- |
| **INSTRUKCIJE** | | **KLJUČNE RIJEČI** | **PRIMJERI** |
| Instrukcije za početak i kraj | | početak, kraj | početak;  kraj; |
| Instrukcije za ulaz i izlaz podataka | | upiši, ispiši | upiši (x);  ispiši (A,B);  ispiši ("Doviđenja");  ispiši (x+y);  ispiši ("Površina je ", P); |
| Instrukcije za obavljanje operacija | | ( = ) | z = x + y;  c = sqrt (a\*a+b\*b);  br = br – 1; |
| Instrukcije za određivanje toka programa | Instrukcije grananja | ako je uvjet onda ... inače ... | ako je x > 0 onda ispiši ("Pozitivan.")  inače ispiši ("Možda negativan.");  ako je b<>0 onda  kvoc = a/b;  ispiši (kvoc)  inače  ispiši ("Ne može se podijeliti."); |
| Instrukcije za ponavljanje | ponoviti broj puta ... | zbroj = 0;  ponoviti 50 puta  ulaz (x)  zbroj = zbroj + x  izlaz (zbroj); |
| Instrukcije za ponavljanje s ispitivanjem uvjeta | dok je uvjet radi ...  *(Napomena: petlja se vrti dok god je uvjet istinit.)* | upiši (x);  zbroj = 0;  i = 1;  dok je i <= x radi  zbroj = zbroj + i;  i = i + 1;  ispiši (zbroj); |
| Instrukcije za ponavljanje s ispitivanjem uvjeta | ponavljati ... do uvjet  *(Napomena: petlja se vrti dok je uvjet lažan, kad postane istinit prelazi se na sljedeću instrukciju.)* | upiši (x);  zbroj = 0;  i = 1;  ponavljati  zbroj = zbroj + 1;  i = i + 1  do i > x;  ispiši (zbroj); |

Podcrtane riječi nazivaju se **ključne** ili **rezervirane riječi**.

One određuju pojedine instrukcije i uvijek se tako pišu. U programskim jezicima prevode se na engleski.  
Niz jednostavnih instrukcija (unutar neke druge instrukcije) može se smatrati jednom složenom instrukcijom koja se označava vertikalnom crtom ispred. Instrukcije se jedna od druge odvajaju znakom ;.

6.5. Dijagram tijeka

Algoritme za rješavanje problema najčešće prikazujemo grafički pomoću dijagrama tijeka.

**Dijagram tijeka** je grafički prikaz algoritma koji olakšava izradu programa.

On nam omogućuje da što jednostavnije i točnije napišemo program i da se u njemu ne pojavljuju greške.

Tako prikazan algoritam vrlo je pregledan i potpuno određen. Posebno je pogodan za analize programa, traženje sličnih rješenja ili traženje potrebnih izmjena.

Pri crtanju dijagrama tijeka služimo se simbolima:

|  |  |
| --- | --- |
|  | početak, kraj i prekid programa |
|  | ulaz podataka |
|  | izlaz podataka |
|  | naredba (operacija) |
|  | grananje (odluka) – uvijek ima jedan "ulaz", a dva "izlaza" |
|  | linije toka programa |

6.6. Primjeri i zadaci

6.6.1. Slijed

Primjer 1: Napiši algoritam i sastavi dijagram tijeka za program koji će učitati polumjer kruga, a izračunati i ispisati promjer, površinu i opseg kruga.

|  |  |
| --- | --- |
| ALGORITAM | DIJAGRAM TIJEKA |
| početak;  upiši R;  D = 2 \* R;  P = R \* R \* 3,14;  O = 2 \* R \* 3,14;  ispiši D, P, O;  kraj | R  D, O, P  D = 2 \* R  P = R \* R \* 3,14  O = 2 \* R \* 3,14 |

**Zadaci:**

1. Napiši algoritam i sastavi dijagram tijeka za računanje zbroja, razlike, produkta i kvocijenta dvaju zadanih brojeva.
2. Napiši algoritam i sastavi dijagram tijeka za računanje opsega i površine pravokutnog trokuta ako su zadane njegove katete.
3. Napiši algoritam i sastavi dijagram tijeka za program koji će računati prosjek četiri zadana broja.
4. Napiši algoritam i sastavi dijagram tijeka za program koji će računati aritmetičku, geometrijsku i harmonijsku sredinu dvaju zadanih brojeva.
5. Napiši algoritam i sastavi dijagram tijeka za program koji će brzinu u m/s pretvarati u km/h.
6. Napiši algoritam i sastavi dijagram tijeka za program koji će vrijeme u satima, minutama i sekundama pretvoriti u vrijeme u sekundama.
7. Napiši algoritam i sastavi dijagram tijeka za program koji će učitati koordinate dviju točaka koord. sustava pa izračunati i ispisati njihovu udaljenost te koordinate polovišta.
8. Napiši algoritam i sastavi dijagram tijeka za program koji će učitati duljinu, visinu i širinu kvadra, a ispisati oplošje i volumen.

6.6.2. Grananje

Primjer 2: Napiši algoritam i sastavi dijagram tijeka za program koji će učitati dva broja pa ispisati koji je veći.

|  |  |
| --- | --- |
| ALGORITAM | DIJAGRAM TIJEKA |
| početak;  upiši A,B;  ako je A>B onda ispiši A  inače ispiši B;  kraj | NE  A, B  B  A > B  A  DA |

**Zadaci:**

1. Napiši algoritam i sastavi dijagram tijeka za program koji će za učitani broj x ispisati da li je pozitivan, negativan ili nula.
2. Napiši algoritam i sastavi dijagram tijeka za program koji će učitati tri broja, provjeriti mogu li oni biti stranice trokuta i ako mogu izračunati i ispisati površinu i opseg tog trokuta.
3. Napiši algoritam i sastavi dijagram tijeka za program koji će učitati dva broja, a ispisati njihov produkt i kvocijent. (Napomena: ako je djelitelj jednak 0, ispiši poruku da se ne može podijeliti.)
4. Napiši algoritam i sastavi dijagram tijeka za program koji će računati apsolutnu vrijednost učitanog broja.
5. Napiši algoritam i sastavi dijagram tijeka za program koji će učitati prosjek ocjena, a zatim ispisati odgovarajući opći uspjeh.
6. Napiši algoritam i sastavi dijagram tijeka za program koji će učitati dva broja i znak za operaciju (+, -, \*, /). U ovisnosti o učitanoj operaciji treba ispisati rezultat ili poruku da je unesen pogrešan znak.
7. Napiši algoritam i sastavi dijagram tijeka za računanje opsega i površine kvadrata ako je zadana duljina stranice kvadrata. Prije računanja provjeriti da li je A>0, a ako nije, ispisati poruku ″Stranica kvadrata mora biti pozitivan broj″.
8. Napiši algoritam i sastavi dijagram tijeka za program koji će računati vrijednost funkcije:  
    x2 , x<0  
    f(x) = 0 , x=0  
    √x , x>0
9. Napiši algoritam i sastavi dijagram tijeka za program koji će učitati dva broja x i y, a zatim ispisati vrijednost funkcije f definirane ovako:

0 , za x=y

f = x2 – y2 , za x>y

y2 – x2 , za y>x

Primjer 3: Napiši algoritam i sastavi dijagram tijeka za program koji će učitati stranice trokuta, a ispisati da li je trokut jednakostraničan, jednakokračan, raznostraničan i/ili pravokutan.

|  |  |
| --- | --- |
| ALGORITAM | DIJAGRAM TIJEKA |
| NE  DA  A, B, C  A\*A+B\*B=C\*C  ili A\*A+C\*C=B\*B ili B\*B+C\*C=A\*A  ″Pravokutan″  ″Nije pravokutan″  A=B i A=C  ″Jednakostraničan″  ″Jednakokračan″  ″Raznostraničan″  DA  A=B ili  A=C ili B=C  NE  DA  NE  početak;  upiši A,B,C;  ako je A\*A+B\*B=C\*C ili A\*A+C\*C=B\*B  ili B\*B+C\*C=A\*A  onda ispiši ″Pravokutan trokut″  inače ispiši ″Nije pravokutan″;  ako je A=B i A=C  onda ispiši ″Jednakostraničan trokut″  inače ako je A=B ili A=C ili B=C  onda ispiši ″Jednakokračan″  inače ispiši ″Raznostraničan″;  kraj |  |

6.6.3. Ponavljanje (petlja)

Primjer 4: Napiši algoritam i sastavi dijagram tijeka za program koji će učitati jedno ime, a zatim 25 puta ispisati da je ″ime″ pametan.

|  |  |
| --- | --- |
| ALGORITAM | DIJAGRAM TIJEKA |
| početak;  upiši IME;  ponoviti 25 puta  ispiši IME, ″ je pametan″;  kraj | IME  IME, ″je pametan″  ponavljaj  25 puta |
| početak;  upiši IME;  BR = 1;  dok je BR<=25 radi  ispiši IME, ″ je pametan″;  BR = BR + 1;  kraj | IME  IME, ″je pametan″  BR = 0  BR<=25  DA  NE |

Primjer 5: Napiši algoritam i sastavi dijagram tijeka za program koji će učitati 50 brojeva i izračunati njihov zbroj.

|  |  |
| --- | --- |
| ALGORITAM | DIJAGRAM TIJEKA |
| početak;  ZBROJ = 0;  ponoviti 50 puta  upiši X;  ZBROJ = ZBROJ + X;  ispiši ZBROJ;  kraj | X  ZBROJ = ZBROJ + X  ZBROJ = 0  ZBROJ  ponavljaj  25 puta  ponavljaj  50 puta |

**Zadaci:**

1. Napiši algoritam za program koji će učitati 50 brojeva, a iza svakog učitanog broja ispisati njegov korijen i kvadrat.
2. Sastavi algoritam i napravi dijagram tijeka za program koji će ispisati kvadrate i korijene brojeva od 1 do 25

BROJ KVADRAT KORIJEN

1 1 1

2 4 1.41

... ... ...

25 625 5.00

1. Sastavi algoritam za program koji će tražiti da se učita jedan broj, a zatim ispisati tablicu množenja tim brojem za prvih 20 prirodnih brojeva.
2. Sastavi algoritam za program koji će učitati 37 brojeva, a izračunati i ispisati njihov produkt.
3. Sastavi algoritam i napravi dijagram tijeka za program koji će za učitani broj x izračunati i ispisati produkt svih brojeva do njega uključujući i njega.
4. Sastavi algoritam i napravi dijagram tijeka za program koji će učitati 100 brojeva, a izračunati i ispisati posebno sumu pozitivnih, a posebno sumu negativnih.
5. Sastavi algoritam i napravi dijagram tijeka za program koji će tražiti da se učita 50 znakova s tipkovnice i ispisati koliko je bilo znakova \*, a zatim još ispisati i toliko znakova \*. (Npr. ako je bilo 7 znakova \*, treba još ispisati \*\*\*\*\*\*\*.)
6. Sastavi algoritam i napravi dijagram tijeka za program koji će učitati broj x, a ispisati sumu definiranu ovako:

SUMA = 12 + 22 + 32 + ... + x2

1. Sastavi algoritam i napravi dijagram tijeka za program koji će učitati brojeve x i y, a ispisati broj z definiran ovako:

z = 1/y + 2/y + 3/y + ... + x/y

1. Sastavi algoritam i napravi dijagram tijeka za program koji će učitati 25 brojeva, a ispisati zbroj njihovih korijena.
2. Sastavi algoritam i napravi dijagram tijeka za program koji će učitati dva broja, a ispisati produkt svih brojeva između njih uključujući i njih.
3. Sastavi algoritam i napravi dijagram tijeka za program koji će učitavati znakove s tipkovnice sve dok se ne učita znak $, a zatim ispisati koliko ukupno znakova je bilo učitano.
4. Sastavi algoritam i napravi dijagram tijeka za program koji će učitati 36 brojeva te ispisati najvećeg među njima i njihov prosjek.

Primjer 6: Napiši algoritam i sastavi dijagram tijeka za program koji će učitati prirodni broj, a izračunati i ispisati zbroj svih brojeva do njega uključujući i njega.

|  |  |
| --- | --- |
| ALGORITAM | DIJAGRAM TIJEKA |
| početak;  upiši N;  ZBROJ = 0;  BR = 1;  dok je BR<=N radi  ZBROJ = ZBROJ + BR;  BR = BR + 1;  ispiši ZBROJ;  kraj | ponavljaj  25 puta  N  ZBROJ = ZBROJ + BR  BR = BR + 1  ZBROJ=0  BR = 1  ZBROJ  DA  BR<=N  NE |

6.7. Faze programiranja

Postupak izrade programa nije nimalo lak posao. Često je potrebno mnogo znanja, snalažljivosti i dobrih ideja da bismo mogli izraditi dobar program. Zato postupak izrade programa dijelimo u nekoliko faza:

**1. RAZUMJEVANJE PROBLEMA**

Da bi uopće mogli krenuti, potrebno je dobro razumijeti zadatak. Moramo znati što se traži u zadatku i predvidjeti sve slučajeve koji se mogu dogoditi pri obradi.

**2. OPIS ULAZNIH I IZLAZNIH PODATAKA**

Moramo točno znati koji podaci su nam zadani (ulazni), kojeg su tipa, postoje li neka ograničenja za ulazne podatke. Također je potrebno znati i što je nepoznato, odnosno koje podatke ćemo odrediti programom (izlazni podaci) i kojeg su tipa.

Ovom se fazom utvrđuje ŠTO se treba raditi a ne kako to učiniti.

**3. ODREDITI ALGORITAM**

Kada točno znamo ŠTO treba raditi, potrebno je odrediti metodu, način rješavanja, odnosno KAKO riješiti zadatak. Tada tražimo dobar, brz i efikasan postupak za rješavanje problema.

Pritom treba paziti na redoslijed izvođenja radnji:

* ulaz podataka
* obrada podataka
* izlaz podataka

**4. NACRTATI DIJAGRAM TIJEKA**

Algoritam kojim ćemo riješiti problem prikazujemo grafički.

**5. PISANJE PROGRAMA U NEKOM PROGRAMSKOM JEZIKU (KODIRANJE)**

Opisani tijek programa sada prevodimo u naredbe nekog višeg programskog jezika, odnosno pišemo program na računalu.

**6. PREVOĐENJE PROGRAMA**

Program treba prevesti iz višeg programskog jezika u strojni jezik kojeg računalo razumije. To rade programi prevoditelji (kompajleri).

**7. IZVRŠAVANJE I TESTIRANJE PROGRAMA**

Zatim po prvi puta možemo pokrenuti program na računalu. Moramo ga isprobati za različite ulazne podatke, birajući ih tako da obuhvatimo najrazličitije mogućnosti ulaznih podataka. Ako naiđemo na neku grešku, tada je ovo faza u kojoj sve greške u programu treba ispraviti.

**8. PISANJE PROGRAMSKE DOKUMENTACIJE**

Nakon što napišemo nekoliko programa, a posebno ako su poduži, nakon nekog vremena nećemo se sjećati koji smo točno postupak koristili za rješavanje kojeg od tih problema. Zato je važno tijekom izrade programa pisati programsku dokumentaciju u kojoj su opisani svi koraci izrade programa.

**9. ODRŽAVANJE PROGRAMA**

Napisani program po potrebi se mora dotjerivati i prilagođavati novim zahtjevima korisnika i novoj opremi.

Autor:

Željka Bjelanović, prof.

Srednja škola Čazma, 2004.

Literatura:

1. Enter, br. 4, Programska oprema osobnog računala – 1. dio, str. 30-34
2. Enter, br. 14, Operativni sustav (1. dio), str 27-32
3. Budin: INFORMATIKA 1 – udžbenik za 1. razred gimnazije, Element, Zagreb, 1996.
4. Grundler: INFORMATIKA 1 – udžbenik za 1. razred opće, jezične i klasične gimnazije, ŠK, Zagreb, 2002.
5. Lipljin i dr.: INFORMATIKA – RAČUNALSTVO za srednje škole – udžbenik za 1. i 2. razred, Prp-mil, Varaždin, 2003.
6. Grabusin – Miletić: C u primjeni – priručnik za gimnazije i srednje tehničke škole, Pentium, Vinkovci, 2003.