2. OSNOVE PROGRAMIRANJA ELEKTRONIČKOG RAČUNALA

2.1. POJAM ALGORITMA

Komunikacija predstavlja razmjenu činjenica ili ideja između ljudi, temelji se na zajedničkom dogovoru o značenju fizičkih simbola (podataka) koji opisuju pojmove, činjenice ili ideje. Informacije predstavljaju značenja pridružena podacima.

Podaci se upotrebljavaju kod:

- prijenosa informacija
- pohrane informacija za buduću upotrebu
- obrade informacija (izvođenja novih informacija na temelju postojećih podataka prema određenim pravilima)

Pravila za rad s podacima nazivamo operacijama. Podaci i operacije su osnovne komponente računanja. Računanje (postupak rješavanja) čini konačan skup operacija primijenjen na konačan skup podataka s ciljem rješavanja postavljenog zadatka

Algoritam je računanje koje rješava postavljeni zadatak. Algoritam treba ispuniti sljedeće uvjete:

- popis operacija (uputa) koje opisuju izvršavanje nekog postupka izražen na jasan i logičan način
- operacije iz popisa (koraci algoritma) moraju biti neposredno izvedive
- nedvosmislen
- završava u konačnom broju koraka

Dobro oblikovani strukturirani algoritam olakšava čitljivost i razumljivost postupka računanja.

2.1.1. OSNOVNE STRUKTURE PRI KONSTRUKCIJI ALGORITMA

Slijedna struktura kod koje se koraci izvršavaju u slijedu jedan iza drugoga, a pojedini korak se izvršava samo jedanput. Izborna struktura izabire se i izvršava samo jedna od ponuđenih mogućnosti. Struktura ponavljanja kod koje se izvođenje jednog ili više koraka se ponavlja.

2.1.2. PRIMJERI JEDNOSTAVNIH ALGORITAMA

1) Zamjena kotača na automobilu:

1. podigni automobil
2. skini kotač
3. ako rezervni kotač nije pripremljen, onda ga pripremi
4. namjesti rezervni kotač
5. spusti automobil

Podjela na jednostavne, lako razumljive i dobro poznate korake ovisi o izvođaču algoritma. Postupak se izvršava propisanim redoslijedom, jedan iza drugoga. Nakon konačnog broja koraka slijedi željeni rezultat.

2) Obračun telefonskih troškova

Zadatak: Sastavite algoritam za izračunavanje telefonskih troškova na kraju mjeseca, ako su poznati broj potrošenih telefonskih impulsa, cijena jednog impulsa te iznos telefonske pretplate. U iznos telefonske pretplate uračunano je 100 impulsa.

Početno poznati podaci:

- broj potrošenih telefonskih impulsa (impulsi)
- cijena jednog impulsa (cijena)
- iznos telefonske pretplate (pretplata)
- broj impulsa uračunat u pretplatu (100)

Željeni rezultantni podatak:

- iznos ukupnih telefonskih troškova (trošak)

Impulsi, cijena, pretplata, iznos su varijable. Varijabla je veličina koja poprima vrijednosti iz skupa dopuštenih vrijednosti. Vrijednost varijable tijekom izvršavanja određuje algoritam. Broje 100 je konstanta. Konstanta je veličina stalne vrijednosti. Zakonitost za izračunavanje troška:

VIŠAK = IMPULSI - 100
TROŠAK = PRETPLATA + VIŠAK * CIJENA

Koraci algoritama:

1. *Unesi podatke*: IMPULSI, CIJENA, PRETPLATA
2. *Ako* IMPULSI > 100 *onda* VIŠAK ← IMPULSI - 100
   inače VIŠAK ← 0
3. TROŠAK ← PRETPLATA + VIŠAK * CIJENA
4. *Ispiši rezultat*: TROŠAK

Objašnjenje koraka algoritma:

1. Pridruživanje vrijednosti varijablama

2. Pravilo zaključivanja:

ako je uvjet zadovoljen	(da/ne)
onda izračunaj	(da)
inače izračunaj	(ne)

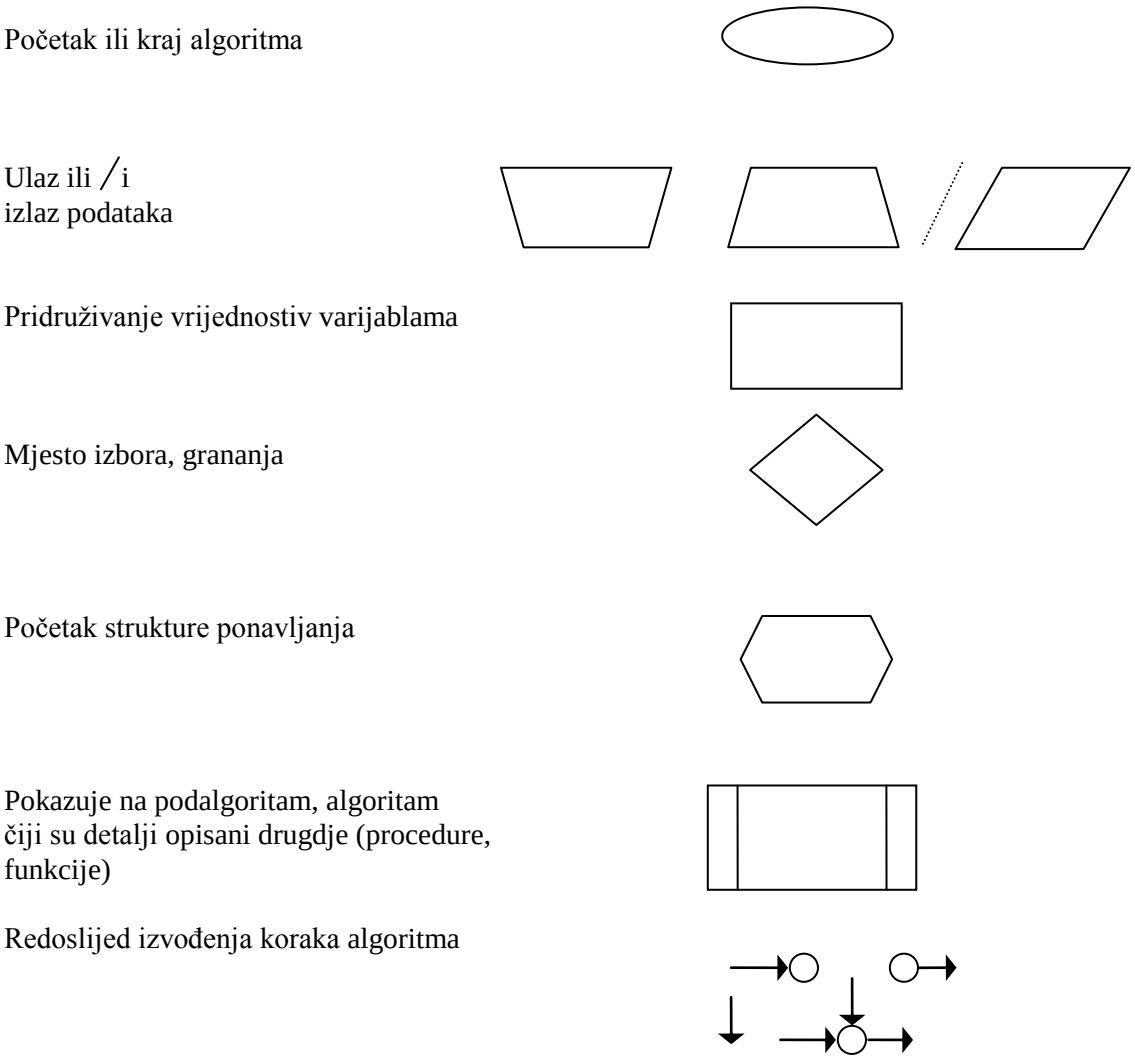
Operacija pridruživanja (←) varijabli s lijeve strane znaka pridruživanja pridružuje se vrijednost izraza s desne strane znaka pridruživanja.

3. Izračunavanje zakonitosti

4. Ispis vrijednosti varijable

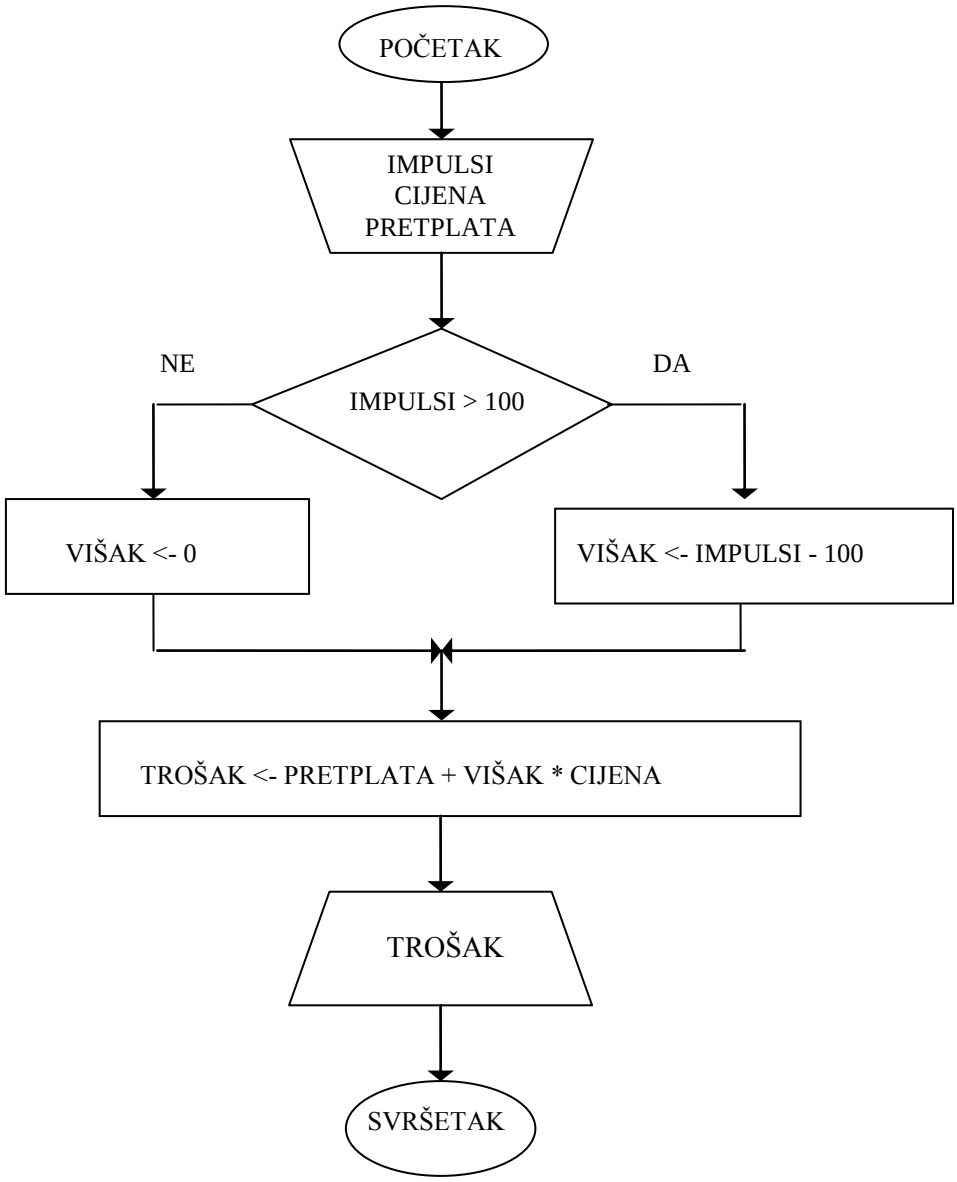
2.1.3. DIJAGRAM TOKA

Dijagram toka pruža pregledniji prikaz algoritma korištenjem standardnih grafičkih simbola, sl. 2.1.



Slika 2.1 Simboli dijagrama toka

Dijagram toka algoritma za obračun troškova telefonske pretplate prikazuje sl. 2.2.



Slika 2.2. Dijagram toka algoritma za obračun telefonskih troškova

2.2. POJAM PROGRAMA

Program je algoritam prilagođen za izvođenje na računalu. Programiranje je pisanje programa.

Postupci u pisanju programa:

1. Kodiranje programa – programer piše program
2. Ispravljanje programa – ispravljanje pogrešaka u programu
3. Testiranje programa – testiranje da bi bio siguran u ispravnost dobivenih informacija – provjera rada prema postavljenim zahtjevima
4. Dokumentiranje programa – opis operacija i uporabe programa
5. Održavanje programa – životni ciklus programa – promjena programa prema zahtjevima korisnika

Programski jezik je sredstvo izražavanja programa. služi za opis zadatka i opis postupka rješavanja. Opis zadatka"što?" je opis polaznih podataka, rezultata i njihovih međusobnih zakonitosti, relacija Opis postupka rješavanja "kako?" je redoslijed primjene zakonitosti, relacija.

2.2.1. STRUKTURIRANO PROGRAMIRANJE

Sustavni pristup programiranju je pristup kod kojeg se program strukturira u dva dijela i to dio programa u kojem se navode, deklariraju korištene varijable i izvedbeni dio.

Deklaracija podataka predstavlja popis varijabli i njihovih tipova. Tip podatka utvrđuje skup dozvoljenih vrijednosti koje mopže poprimiti varijabla.

Izvedbeni dio odnosno redoslijed izvođenja naredbi programa promatra se preko koraka algoritma.

Struktura programa izražena programskim jezikom PASCAL glasi:

```
program .....          - zaglavlje programa
.....
.....                  - deklaracija podataka
.....
begin (početak)
    .....
    .....              - izvedbeni dio
    .....
end.  (svršetak)
```

Algoritam za obračun telefonskih troškova zapisan u programskom jeziku PASCAL, sl. 3.3.

```
program Telefon;
    var impulsi, visak: integer;
    pretplata, cijena, trosak: real;
begin
    readln(impulsi, cijena, pretplata);
    if impulsi > 100 then       visak := impulsi - 100
                                else visak := 0;
    trosak := pretplata + visak * cijena;
    writeln(' Trosak iznosi ', trosak:6:2, ' kn')
end.
```

Slika 2.3. Algoritam za obračun telefonskih troškova u programskom jeziku PASCAL

Komentar: Iskazi, naredbe u programskom jeziku PASCAL se završavaju znakom točka zarez.

var *naziv: tip*;

integer - varijable impulsi i VISAK poprimaju vrijednosti iz skupa cijelih brojeva (integer)

real - varijable PRETPLATA, CIJENA I TROSAK poprimaju vrijednosti iz skupa decimalnih, realnih brojeva (real)

readln(.....) - pročitaj redak vrijednosti s ulazne jedinice računala (tipkovnica, datoteka) i pridruži ih varijablama navedenima unutar zagrada

if ... **then** ... **else** - struktura uvjeta
ako ... onda ... inače

:= - operacija pridruživanja

+, - , *, / - aritmetički operatori

writeln(.....) - ispiši tekst komentara i vrijednosti varijabli na standardnu izlaznu jedinicu računala (zaslon, datoteka, tiskalo)

' ..... ' - zapis teksta koji će biti neposredno ispisan

varijabla:n:m - format ispisa varijable:
n = ukupni broj mjesta za ispis
m = broj decimalnih mjesta

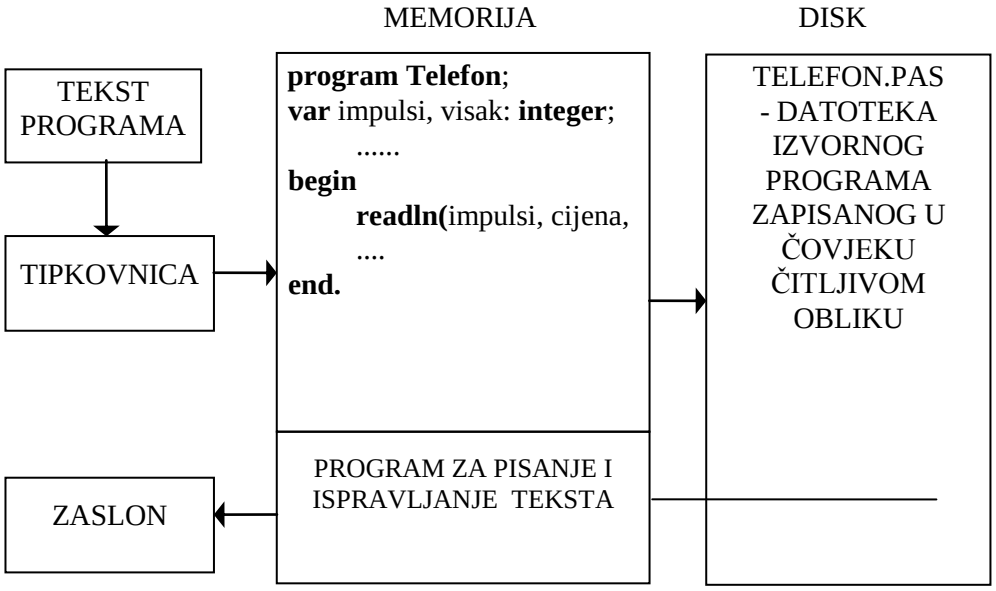
2.3. PRIPREMA PROGRAMA ZA IZVOĐENJE

1. Unos izvornog programa u računalo
2. Prevođenje i tvorba izvedbenog oblika
3. Izvođenje programa

2.3.1. UNOS IZVORNOG PROGRAMA U RAČUNALO

Prijenos teksta, izvornog oblika, programa s papirnatog medija u memoriju računala te njegova pohrana u datoteku na magnetskom mediju (disk) pomoću programa za uređivanje teksta.

Datoteka je imenovani skup povezanih podataka koji čine logičnu cjelinu (program, mjerni podaci, dokument, slika, zvučni zapis, video zapis).

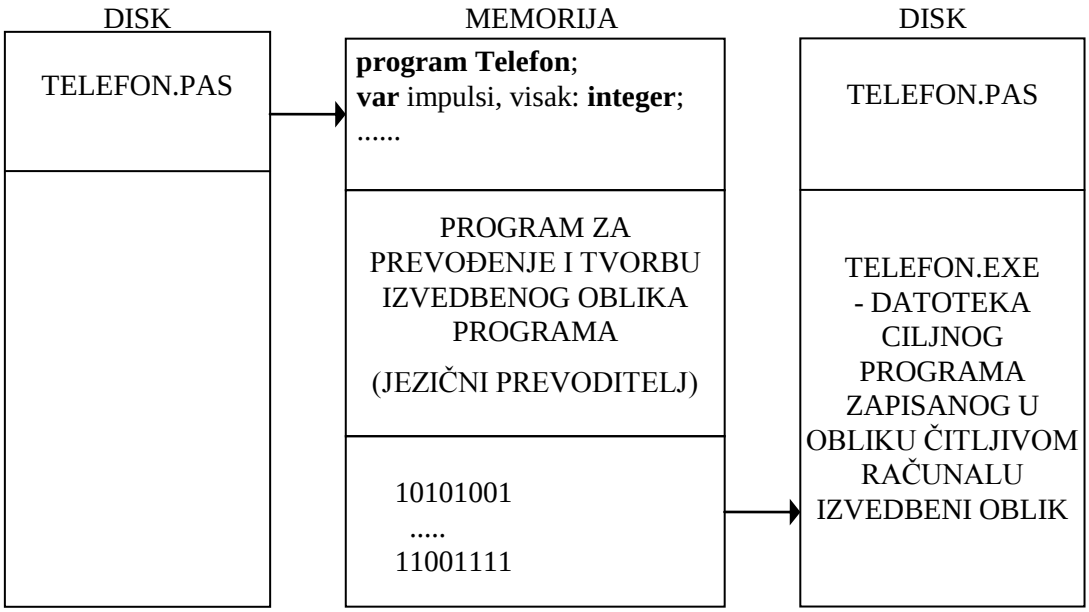


Slika 2.4 Unos programa u računalo

2.3.2. PREVOĐENJE I TVORBA IZVEDBENOG OBLIKA

Program izražen programskim jezikom u izvornom obliku čitljivom čovjeku program jezičnog prevoditelja prevodi u ciljni, stroju čitljivi - čovjeku gotovo nečitljivi, izvedbeni oblik, sl. 2.5.

Izbor programa za prevođenje ovisi o programskom jeziku u kojem je zapisan izvorni program.



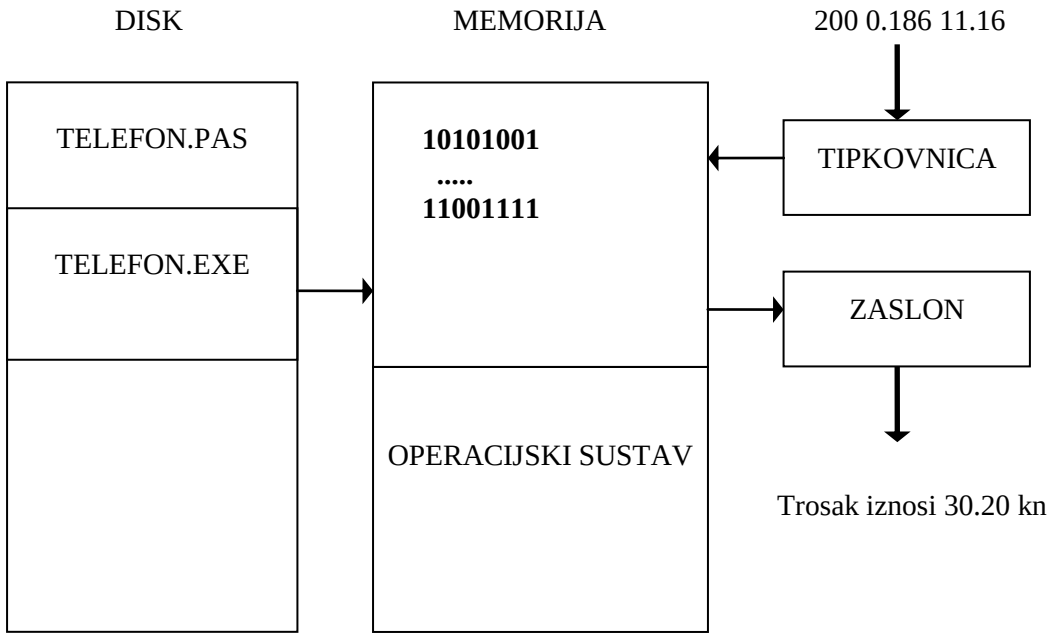
Slika 2.5. Prevođenje i tvorba izvedbenog oblika programa

2.3.3. IZVOĐENJE PROGRAMA

Punjenje izvedbenog oblika programa u memoriju i pokretanje programa pomoću upravljačkih naredbi operacijskog sustava računala.

Operacijski sustav je program za upravljanje radom računala, nadzor programsko-sklopovskih sredstava računala i komunikaciju s čovjekom.

Nakon pokretanja programa unose se putem ulazne jedinice početno poznati podaci, a računalo na temelju programa izračunava rezultantnu vrijednost koju ispisuje na izlaznoj jedinici računala.



Slika 2.6. Izvođenje programa

Algoritam za obračun telefonskih troškova zapisan u programskom jeziku C:

```
main()
{
  int impulsi, visak;
  float pretplata, cijena, trosak;
      scanf("%f %d %f", &pretplata, &impulsi, &cijena);
          if (impulsi > 100)
                  visak = impulsi - 100;
          else visak = 0;
      trosak = pretplata + visak*cijena;
      printf("Trosak iznosi %f kn", trosak);
}
```

2.4. OSNOVE PROGRAMIRANJA

Program je postupak rješavanja zadatka prilagođen izvođenju na računalu. Osnove programiranja izložene su pomoću programskog jezika PASCAL.

2.4.1. ELEMENTI PROGRAMA

Ulazna operacija služe za unos vrijednosti s ulazne jedinice (tipkovnica, disk, ulazna vrata) u program

Podatak u programu može biti konstanta, varijabla i struktura. Podatak čuva vrijednosti brojeva (cijeli, decimalni), tekstova (znakove, karaktere i nizove) ili adresa (varijable ili strukture) na određenom mjestu u memoriji (memorijskoj lokaciji).

Operacije u programu rade s podacima pomoću naredbi, instrukcija koje pridružuju vrijednosti, slažu vrijednosti (zbrajanje, dijeljenje, i sl.) i uspoređuju vrijednosti (jednako, različito i sl.).

Izlazna operacija služi za iznos vrijednosti iz programa, npr. ispis na zaslon izlazne jedinice, tiskalo (riječi i slike), memorijsku jedinicu (disketa ili disk) ili izlazna vrata.

2.4.2. ORGANIZACIJA PROGRAMA

Program može biti organiziran za uvjetno izvođenje, ponavljanje naredbi ili u obliku potprograma. U slučaju uvjetnog izvođenja instrukcija ili skup instrukcija se izvodi samo kada je ispunjen određeni uvjet ili niz uvjeta. Petlja - izvođenje instrukcije ili skupa instrukcija ponavlja se više puta

Potprogram, procedura, funkcija - instrukcije se razlučuju u skupove kojima se pridružuje naziv, a izvode se pozivom odgovarajućeg skupa

2.4.3. LOGIČKA ORGANIZACIJA MEMORIJE

Memoriju tvori niz adresibilnih memorijskih lokacija određene duljine. Adresa memorije odgovara nazivu podatka (varijable, strukture). U memorijske lokacije pohranjuju se podaci:

naredbe programa, vrijednosti, adrese drugih lokacija. Duljina memorijske lokacije izražena je osnovnom informacijskom jedinicom (bajt).

2.4.4. TIPOVI PODATAKA I OPERACIJE S PODACIMA

Tipovi podataka

Cijeli broj: pohrana cjelobrojnih vrijednosti u području od najmanje negativne vrijednosti do najveće pozitivne vrijednosti. Područje vrijednosti ovisi o upotrijebljenom jezičnom prevoditelju i za osnovne cjelobrojne tipove podataka prikazano je u tablici 2.1.

Tablica 2.1. Područje vrijednosti cjelobrojnih tipova podataka

TIP	PODRUČJE VRIJEDNOSTI	VELIČINA U BAJTIMA
byte (bajt)	0..255	1
shortint (kratki c.b.)	-128..127	1
integer (c.b.)	-32768..32767	2
word (riječ)	0..65535	2
longint (dugi c.b.)	-2147483648..2147483647	4

Realni broj: pohrana u formatu pokretnog zareza sastavljenom od mantise (vrijednosti decimalnog broja) i eksponenta koji određuje mjesto zareza. Broj znamenaka mantise (značajne znamenke) i područje vrijednosti eksponenta ovise o korištenom jezičnom prevoditelju. Područje vrijednosti ovisi o upotrijebljenom jezičnom prevoditelju i za osnovne cjelobrojne tipove podataka prikazano je u tablici 2.2.

Tablica 2.2. Područje vrijednosti realnih tipova podataka

TIP	PODRUČJE VRIJEDNOSTI	ZNAČAJNE ZNAMENKE	VELIČINA U BAJTIMA
real (decimalni broj)	$10^{-38}..10^{38}$	11	6
single (jednostruki)	$10^{-38}..10^{38}$	7	4
double (dvostruki)	$10^{-308}..10^{308}$	15	8

Znak, karakter: pohrana slova abecede (a,b,...., a, b, ...), znamenki (od 0 do 9), simbola (? , ! . :). Pohrana jednog slova u jednom bajtu. Znakovne konstante se prikazuju unutar navodnika ('a', 'e', '?', '2'). Više znakova čini znakovni niz – string ('osnove programiranja').

Booleove konstante imaju vrijednost istinitosti je istina ili laž.

Kazalo služi za pohranu adrese memorijske lokacije, a ne njezinog sadržaja.

Identifikatori su nazivi konstanti, tipova podataka, varijabli, funkcija.

integer, real	tipovi podataka određeni programskim jezikom
Telefon	naziv glavne funkcije programa i nazive
impulsi, cijena	varijabli zadaje programer
readln, writeln	procedure određene programskim jezikom

Prvi znak identifikatora mora biti slovo. Dopušteni broj znakova identifikatora određuje jezični prevoditelj. Nazivi tipova podataka, naredbi i procedura su rezervirane, ključne riječi programskog jezika.

Operatori

Rukovanje vrijednostima unesenima u program, odnosno pohranjenima u varijablama - tvorba izraza - izvođenje s lijeva na desno. Binarni (djeluju na dva operanda), unarni (djeluju na jednog operanda). Pridruživanja, usmjereni na rukovanje s bitovima, relacijski, logički, adresni, niza.

Operator pridruživanja

Vrijednost rezultata izraza s desne strane operatora pridruživanja pridružuje se varijabli s lijeve strane operatora pridruživanja.

Aritmetički operatori primjenjuju se na podatke cjelobrojnog i realnog tipa i prikazani su u tablici 2.3.

Tablica 2.3. Aritmetički operatori

Zbrajanje	+	7 + 2 = 9
Oduzimanje	-	7 - 2 = 5
Množenje	*	7 * 2 = 14
Dijeljenje	/	7 / 2 = 3.5
Cjelobrojno dijeljenje	div	7 div 2 = 3
Ostatak dijeljenja	mod	7 mod 2 = 1

Operatori usmjereni na rukovanje s bitovima

Bit je najmanja informacijska jedinica, vrijednost 0 ili 1, pohrana cjelobrojnim tipom podatka. 8 bita je 1 bajt.

Pomak bitova za određeni broj pozicija na lijevu stranu, desna strana se zamjenjuje s 0 - (shl - shift left).

Pomak bitova za određeni broj pozicija na desnu stranu, lijeva strana se zamjenjuje s 0 - (shr - shift right)

Izvođenje logičke operacije “i” nad odgovarajućim parom bitova. Rezultantna vrijednost je 1 ako oba bita u paru imaju vrijednost 1, u protivnom rezultat je 0 - (and)

Izvođenje logičke operacije “ili” (uključivo ili) nad odgovarajućim parom bitova. Rezultantna vrijednost je 0 ako oba bita u paru imaju vrijednost 0, u protivnom rezultat je 1 - (or)

Izvođenje logičke operacije “isključivo ili” nad odgovarajućim parom bitova. Rezultantna vrijednost je 1 ako su vrijednosti bitova u paru različite, u protivnom rezultat je 0 - (xor)

Logičko komplementiranje, negacija bita, promjena 0 u 1 i 1 u 0 - (not)

Relacijski operatori

Relacijski operatori služe za uspoređivanje dvije vrijednosti. Rezultat usporedbe je booleova vrijednost (istina, laž). Relacijski operatori korišteni u programskim jezicima prikazani su u tablici 2.4.

Tablica 2.4. Relacijski operatori

Veće	>	7 > 2 = istina
Veće ili jednako	>=	7 >= 2 = istina
Manje	<	7 < 2 = laž
Manje ili jednako	<=	7 <= 2 = laž
Jednako	=	7 = 2 = laž
Različito	<>	7 <> 2 = istina

Logički operatori su operatori "I", "ILI", "ISKLJUČIVO ILI", "NE" i su slični operatorima usmjerenima na rukovanje s bitovima samo što rade s booleovim vrijednostima. Tablice istinitosti za navedene logičke operatore prikazane su u tablicama od 2.5. do 2.7.

Tablica 2.5. Tablica istinitosti operatora "I" - Z = A AND B

A	*B*	*Z*
laž	laž	laž
laž	istina	laž
istina	laž	laž
istina	istina	istina

Tablica 2.6. Tablica istinitosti operatora "ILI" - Z = A OR B

A	*B*	*Z*
laž	laž	laž
laž	istina	istina
istina	laž	istina
istina	istina	istina

Tablica 2.7. Tablica istinitosti operatora "NE" - Z = NOT A

A	*Z*
laž	istina
istina	laž

Adresni operator

Rezultat primjene adresnog operatora je adresa memorijske lokacije varijable (@). Rezultat je vrijednost pohranjena u memorijskoj lokaciji na adresi na koju pokazuje kazalo (^).

Operatori niza izvode stapanje dvaju znakovnih nizova (+).

Prioritet primjene operatora prikazan je tablicom 2.8. Ako postoje, zagrade određuju prioritete primjene operatora.

Tablica 2.8. Prioriteti primjene operatora

Operatori	Kategorija	Prioritet
@, not	unarni operatori	najviši
*, /, div, mod, and, shl, shr	operatori množenja	
+, -, or, xor	operatori zbrajanja	
=, <>, <, >, <=, >=,	relacijski operatori	najniži

Komentari predstavljaju napomene programera u programu (značenja varijabli, instrukcija, funkcija), npr. (* .... *).

2.4.5. ULAZNO-IZLAZNE OPERACIJE

Izlazna operacija služi za ispis podataka na izlaznu jedinicu (zaslon, tiskalo, disk).

- **Writeln**(var1, var2, ...);

var = varijabla čija se vrijednost želi ispisati vrijednosti varijabli se ispisuju prema zapisanom redoslijedu u jednom redu, a na kraju ispisa prelazi se na početak novog reda

- **Write**(var1, var2, ...);

Vrijednosti varijabli se ispisuju prema zapisanom redoslijedu u jednom redu, a na kraju ispisa ostaje se u istom redu

- **Writeln**(var1,' ',var2,' ', ...);

Bjeline između članova treba posebno navesti. Primjer ispisa vrijednosti varijable:

```
A := 1; B := 2; C := 3;
Predmet := 'racunalo';
Writeln(A, B, C);                    123
Writeln(A,' ',B,' ',C);              1 2 3
Writeln('Malo',Predmet);             Maloracunalo
Writeln('Malo ',Predmet,'.');        Malo racunalo.
```

- **Writeln(var:mjesta, ...);**

mjesta = ukupni broj mjesta za ispis vrijednosti, vrijednosti se pozicioniraju udesno

Primjer formatiranog ispisa vrijednosti varijable:

```
A := 10; B := 2; C := 100;
Writeln(A, B, C);                102100
Writeln(A:2,B:2,C:3);            10 2100
Writeln(A:3,B:2,C:3);             10  2100
Writeln(A,B:2,C:4);              10 2 100

X := 421.53;
Writeln(X);                       4.2153000000E+02
Writeln(x:8);                     4.21E+02
```

Realni brojevi se ispisuju u eksponencijalnom obliku

- **Writeln**(var:mjesta:decimale, ...);

decimale = broj mjesta za ispis decimalnog dijela vrijednosti realnog broja

Primjer ispisa vrijednosti varijable X := 421.53.

```
Writeln(X:6:2);                 421.53
Writeln(X:8:2);                 421.53
Writeln(X:8:4);               421.5300
```

Ulazna operacija služi za unos podataka s ulazne jedinice (tipkovnica, disk). podaci se odvajaju bjelinama.

Read(var1,var2, ...); pridruživanje vrijednosti varijablama u popisu
Readln(var1,var2, ...); pridruživanje vrijednosti varijablama u popisu s pomakom u novi red

var = varijabla određenog tipa

2.4.6. UPRAVLJANJE IZVOĐENJEM PROGRAMA

Uvjetno izvođenje naredbi je slučaj kada skup naredbi u programu se izvodi samo ako je ispunjen postavljeni uvjet.

Naredba ako (**if**) predstavlja naredbu izbora

- **if** izraz
  then naredba1
  else naredba2

izraz = booleov izraz – vrijednost izraza je istina (da) ili laž (ne)

Ako je izraz istinit, izvodi se naredba1, u protivnom naredba2

- **if** izraz
  then naredba1

Ako je izraz lažan, naredba1 se ne izvodi

- **if** izraz
  then naredba
  else begin
  naredba1;
.....
  naredban
  end;

Složena naredba

begin naredba1, naredba2, ... naredban **end**;

Naredba slučaj (case) - naredba izbora

```
case izbornik_slučaja of
    oznaka_slučaja1 : naredba1;
    oznaka_slučaja2 : naredba2;
                      .....
    oznaka_slučajan : naredban;
 end;
```

Tip podatka oznaka_slučaja mora biti jednak tipu podatka izbornik_slučaja. Oznaka slučaja može poprimiti vrijednosti jedna ili više konstanti ili područje vrijednosti - dvije konstante odvojene znakom '..'.

Izvodi se naredba čija je oznaka slučaja jednaka tekućoj vrijednosti izbornika slučaja ili se nalazi unutar područja vrijednosti. Ako niti jedna oznaka slučaja ne sadrži vrijednost izbornika slučaja, ne izvodi se niti jedna naredba.

Primjer naredbe slučaja:

```
case Smjer of
   'c', 'c7'     : cest := cest + 1;
   'v', 'v7'     : vodni := vodni + 1;
   'tt'    : telekom := telekom+ 1;
   'pt', 'p7'    : post := post + 1;
   'st', 'z7'    : zeljez := zeljez + 1;
   'a', 'a7'     : zracni := zracni + 1
end;
```

Naredba petlje se koristi u slučaju kada se izvođenje naredbi ponavlja.

Naredba **while**

while uvjet **do** naredba;

uvjet = booleov izraz
naredba = jednostavna ili složena naredba

Izvođenje naredbe se ponavlja sve dok je uvjet zadovoljen (while). Ako kod ulaza u naredbu while uvjet nije zadovoljen, naredba se ni jedanput ne izvodi.

Primjer naredbe slučaja

```
program Ponavljaj;
var
        A, B, Broj      : integer;
        Test            : boolean;
begin
        Write('Upisite dva broja: ');
        Readln(A, B);
        Test := A > B;
        Broj := 1;
        while Broj <= 5 do begin
                Writeln('A je vece od B ', Test);
                Broj := Broj + 1;
        end;
        Writeln('Svrsetak ispisa!')
end.
```

Naredba **for**

for indeks := izraz1 **to** izraz2 **do** naredba;

indeks = indeksna varijabla skalarnog tipa
(cijeli broj, realni broj, znak, booleova vrijednost)

izraz1 = početna vrijednost indeksne varijable

izraz2 = konačna vrijednost indeksne varijable

naredba = jednostavna ili složena naredba

Broj izvođenja naredbe određen je vrijednošću indeksne varijable (indeks) koja se mijenja od početne (izraz1) do konačne (izraz2) vrijednosti s korakom jedan.

to = povećanje indeksa za 1 - inkrementiranje

downto = smanjenje indeksa za 1 - dekrementiranje

Primjer korištenja naredbe for kod zadatka izračunavanja faktorijela

```
F = N! = 1 * 2 * 3 * ..... * N
F := 1;
for i := 1 to N do
        F := F * i;
```

Naredba **repeat**

repeat naredba **until** uvjet;

uvjet = booleov izraz
naredba = jednostavna ili složena naredba

Izvođenje naredbe se ponavlja sve dok uvjet nije zadovoljen (until). Naredba se izvodi barem jedanput jer se uvjet ispituje na kraju.

PRIMJER:

```
program Dijeli;
var     A, B : integer;
        Kolicnik : real;
        Odgovor : char;
begin
        repeat
         Write('Upisite dva broja: ');
         Readln(A, B);
         Kolicnik := A / B;
         Writeln('Kolicnik: ', Kolicnik);
         Write('Zelite li zadati nove brojeve ? (D/N) ');
         Readln(Odgovor)
        until (Odgovor = 'n') or (Odgovor = 'N')
end.
```

2.4.7. POTPROGRAMI

Primjena istog skupa naredbi na različitim skupovima podataka ili na različitim mjestima u programu. Definiraju se na početku programa iza deklaracije varijabli.

- Funkcija - uz zadane vrijednosti argumenta vraća izračunatu vrijednost u program

```
x := Abs(a);
```

- Procedura - se poziva kada se želi izvesti jedan ili više zadataka, bez vraćanja rezultata

```
Writeln('Ovo je pokus');
```

Formalni parametri navode se kod definicije potprograma. Stvarni parametri zamjenjuju formalne kod stvarnog računanja.

Struktura definicije funkcije:

```
function NazivFunkcije(formalni parametri) : tip rezultata;
const
        navođenje konstanti;
type
        definicija tipova podataka;
var
        navođenje varijabli;
        procedure i funkcije;
begin
        tijelo funkcije;
end;
```

Struktura definicije procedure:

```
procedure NazivProcedure(formalni parametri);
const
        navođenje konstanti;
type
        definicija tipova podataka;
var
        navođenje varijabli;
        procedure i funkcije;
begin
        tijelo procedure;
end;
```

Struktura programa:

```
program NazivPrograma;
const
        navođenje konstanti;
type
        definicija tipova podataka;
var
        navođenje varijabli;
        definicija procedura i funkcija;
begin
        glavno tijelo programa
end.
```

Razlika strukture programa i potprograma:

- zaglavlje (program, function, procedure)
- svršetak programa je točka, a potprograma točka zarez

Primjer programa za dijeljenje dva broja:

```
program Dijeljenje;
var A, B              : integer;
    Kolicnik          : real;
 (* Definicija procedure Upis *)
procedure Upis(var X, Y : integer);
(* X, Y Formalni parametri *)
(* var - Stvarni parametri moraju biti varijable - Poziv po imenu*)
begin
      Write('Zadajte dva broja: ');
      Readln(X, Y)
end;
function Dioba(I, J : integer) : real;
begin
      Dioba := I/J
end;
begin
      (* Poziv procedure Upis *)
      Upis(A, B);    (* A, B - Stvarni parametri *)
      (* Poziv funkcije Dioba *)
      (* Stvarni parametri ne moraju biti varijable - Poziv po vrijednosti*)
      Kolicnik := Dioba(A, B);
      Writeln('Kolicnik iznosi ', Kolicnik)
end.
```

2.5. PROGRAMSKI JEZICI

Jezici se dijele na prirodne i umjetne jezike. Umjetni jezici su npr. esperanto, programski jezici. Programski jezici se dijele na više i niže programske jezike.

Viši programski jezici kod kojih je opis zadatka i opis postupka za rješavanje zadatka bliži ljudskom načinu opisa. Zavisno o načinu izvođenja dijele se na proceduralne i neproceduralne programske jezike.

Proceduralno, algoritamski usmjereni su izražajna sredstva za opisivanje podataka i za konstrukciju algoritama: ADA, APL, BASIC, C, COBOL, FORTRAN, LISP, LOGO, PASCAL, PL/I.

Neproceduralno usmjereni su izražajna sredstva za opisivanje podataka i njihovih međusobnih relacija. Kod te skupine programskih jezika programska sredina pronalazi niz operacija ili postupke kojim se podaci, zadovoljenjem njihovih međusobnih relacija, prevode u rezultate: prolog

Niži programski jezici kod kojih je opis zadatka i opis postupka za rješavanje zadatka bliži građi i načinu izvođenja na računalu. Strojni jezik kod kojeg je programe zapisane u strojnom jeziku (binarnom obliku), tj. neposredne upute za rad, moguće koristiti samo na računalu određene arhitekture.