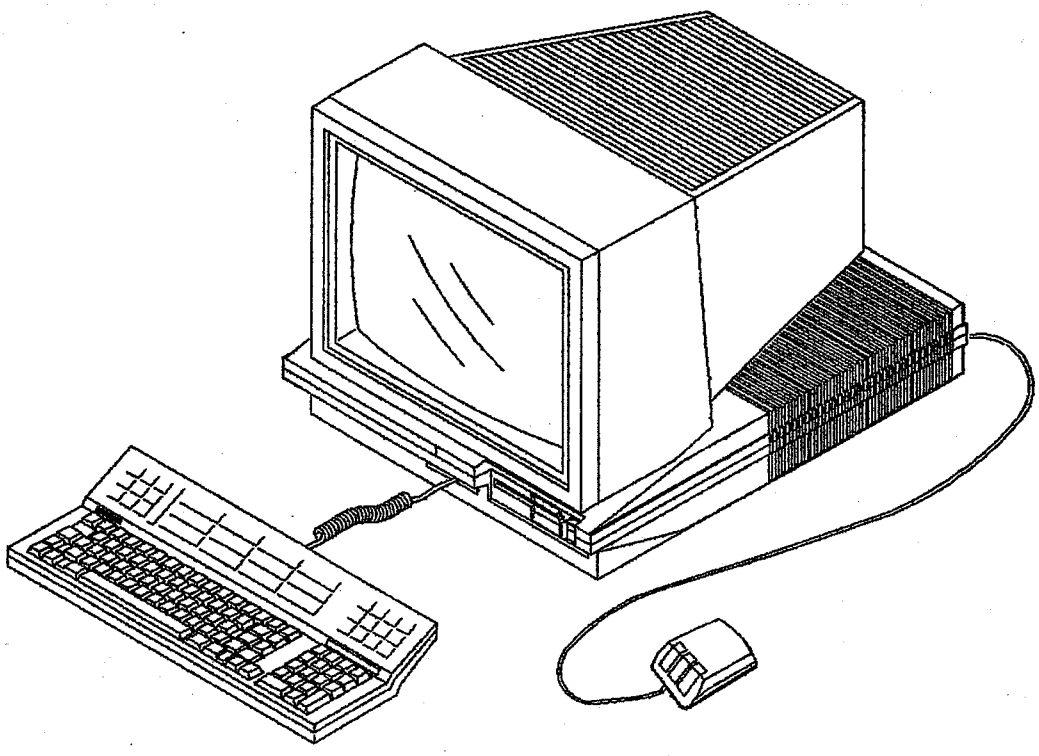
MILE PAVLIC

UVOD U

FORTRAN 77

ZA VELIKA I PC RACUNALA



U V O D U F O R T R A N 7 7

MILE PAVLIC

RIJEKA,1986.

S A D R Z A J:

Predgovor
1. Uvod 1
2. Elementarne konstrukcije 6
2.1. Simboli 6
2.2. Konstante 7
2.2.1. Numericke konstante 7
2.2.2. Literal konstante 9
2.2.3. Logicke konstante 9
2.3. Varijable 9
2.3.1. Numericke varijable 10
2.3.2. Literalne varijable 11
2.3.3. Logicke varijable 11
2.4. Implicitno odredjivanje tipa varijable 11
2.5. Matrice 12
2.6. Izrazi 15
2.6.1. Aritmeticki izrazi 15
2.6.2. Literal izrazi 17
2.6.3. Logicki i relacijski izrazi 18
2.7. Hijerarhija operatora 21
3. Program 22
3.1. Nacin pisanja programa 23
3.2. Dijagram toka programa 24
3.3. Izvrsenje FORTRAN programa na terminalu 27
3.4. Naredbe 28
4. Naredbe za dodijeljivanje vrijednosti 29
4.1. Aritmeticke naredbe 29
4.2. Literal naredbe 31
4.3. Logicke naredbe 32
5. Kontrolne naredbe 33
5.1. END naredba 33
5.2. STOP naredba 35
5.3. PAUSE naredba 35
5.4. CONTINUE naredba 36
5.5. Naredbe za bezuvjetni prelazak 36
5.5.1. Bezuvjetni GO TO 36
5.5.2. Izracunati GO TO 37
5.6. Naredbe za uvjetni prelazak 39
5.6.1. Aritmeticka naredba grananja 39
5.6.2. Logicka naredba grananja 39
5.6.3. Blok naredba grananja 41
5.7. DO naredba 46
5.8. Dodijeljivanje pocetnih vrijednosti 49
5.9. DATA naredba 50
6. Ulazno/Izlazne naredbe 52
6.1. Opcenito 52
6.2. READ naredba 54
6.3. WRITE naredba 56
6.4. FORMAT naredba 56
6.5. Opisivaci polja 57
6.5.1. INTEGER opisivac 58
6.5.2. Drugi oblik INTEGER opisivaca 59
6.5.3. REAL opisivac s fiksnim zarezom 59
6.5.4. REAL opisivac s pokretnim zarezom 60
6.5.5. Specijalni REAL opisivac s pomicnim zarezom ... 61
6.5.6. DOUBLE PRECISION opisivac s pomicnim zarezom .. 61
6.5.7. Tretiranje praznih pozicija 61
6.5.8. Ispis predznaka plus ("+") 61
6.5.9. Logicki opisivac 61
6.5.10. Alfanumericki opisivac 62
6.5.11. Hollerith opisivac 62
6.5.12. Literal opisivac 63

6.5.13. Opisivac za skok 63
6.5.14. Pozicioniranje na mjesto u slogu 63
6.5.15. Prelaz na novi slog 64
6.5.16. Kontrola stampe 64
6.5.17. Ponavljanje grupe opisivaca 65
6.5.18. Zadaci 68
6.6. Datoteka 72
6.7. BACKSPACE naredba 74
6.8. ENDFILE naredba 75
6.9. REWIND naredba 75
6.10. OPEN naredba 76
6.11. CLOSE naredba 76
6.12. Naredba READ za direktni pristup 77
6.13. Naredba WRITE za direktni pristup 78
6.14. Naredba READ za unutrasnje citanje 78
6.15. Naredba WRITE za unutrasnje citanje 79
6.16. Istrazivanje datoteka 80
6.17. Citanje i pisanje matrica 80
7. Specifikacione naredbe 83
7.1. EQUIVALENCE naredba 83
7.2. COMMON naredba 85
7.3. PARAMETAR naredba 86
8. Potprogrami 87
8.1. PROGRAM naredba 90
8.2. Naredba za definiranje funkcija 90
8.3. Poziv potprograma 91
8.4. Funkcijski potprogram 92
8.5. RETURN naredba 93
8.6. SUBROUTINE naredba 94
8.7. ENTRY naredba 96
8.8. BLOCK DATA potprogram 97
8.9. SAVE naredba 97
9. Zadaci 98

Dodaci:

A. ASCII karakteri 102
B. Fortran naredbe 103
C. Standardne fortranske funkcije 104
D. Ogranicenja na poredak naredbi u programskoj jedinici . 106
Literatura 107
Index 108

P R E D G O V O R

U ovom prirucniku je izlozen jedan provjereni i dosad najsire upotrebljavani programski jezik, FORTRAN, i to njegov posljednji medjunarodni standard FORTRAN 77.

Cilj prirucnika je da posluzi kao ucilo pocetnicima (bilo u skolstvu bilo u privredi), da u brzom ucenju savladaju FORTRAN, odnosno sintaksu i njeno koristenje. Mnogim koji dolaze u mogucnost da sa FORTRAN-om 77 rade na PC ili velikim racunarima a inace zapocinju s mukotrpnim ucenjem iz prirucnika proizvodjaca omoguciti ce se da se osamostale i prirucnik proizvodjaca koriste samo kao podsjetnik formata naredbi. Prakticarima FORTRANA IV i FORTRANA V biti ce od pomoci za prelazak na visi nivo.

Prirucnik je nastao prakticnom upotrebom jezika i u obrazovanju kadrova za njegovu upotrebu.

Osnovni nedostaci prirucnika su:

- Sva ogranicenja mogucnosti jezika, koja nisu propisana standardom, a nuzna su i definirana od svakog pojedinog proizvodjaca, data su za UNIVAC 1100 serije, te ih prakticar na drugom racunaru ukoliko ima poteskoca mora procitati u prirucniku proizvodjaca.
- Koristeni su samo tradicionalni dijagrami toka programa za prikaz logike procesa, mada postoje i druga sredstva sve cesce koristena
- Problemi razvoja programskih sistema, kao niza slozenih programa sa znatnom medjusobnom povezanoscu, pomocu savremenijih metoda i tehnika projektiranja programa, dio je posla koji je neobradjen i prepusten programerima. Ovaj vazan posao se moze smatrati visim tecajem programiranja.

U prva dva poglavlja dani su opci pojmovi potrebni za programiranje a od treceg poglavlja nadalje postupno se izucavaju naredbe i vjezba programiranje.

Redoslijed naredbi je pazljivo izabran tako da prethode logicki jednostavnije i potrebnije za razvoj logike programiranja a sve u okviru iste grupe naredbi. Grupa ulazno/izlaznih naredbi je u sestom poglavlju, kako bi se najprije jasno sagledale mogucnosti jezika pa tek onda do u detalje izucavale ove sintaksno slozenije naredbe.

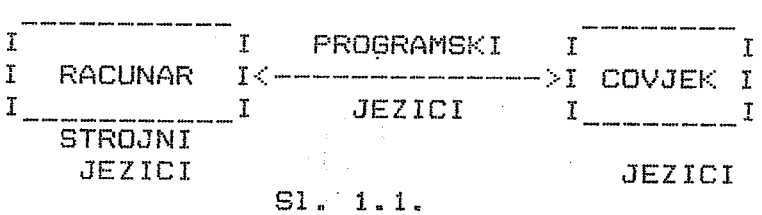
Zahvaljujem se svima koji su mi pomogli u izradi prirucnika.

Autor

1. U V O D

Visi programsk jezici su alat (sredstvo) za brze i lakse kreiranje i dokumentiranje efikasnih i pouzdanih programa. Namjenjeni su za komuniciranje izmedju covjeka i racunara.

Svrha programskih jezika:



Sl. 1.1.

Pisuci program (programirajuci), u visim programskim jezicima (VPJ), za razliku od jezika nizeg nivoa (asemblera, strojnog jezika) olaksava se posao programeru jer on ne mora da:
- poznaje detalje racunara za koga pise program
- poznaje osnovne funkcije koje izvrsava racunar

Pri rijesavanju nekog problema VPJ omogucuju programeru usmijeravanje misli, koristeci sintaksu jezika, tako da na prirodan i egzaktan nacin postavlja algoritam koji vodi k rjesenju. Sintaksa jezika (naredbe racunaru) bliske su govornom jeziku (engleskom).

Sintaksa programskog jezika ne ovisi o racunaru, te se program napisan za rjesavanje nekog problema s jednog racunara moze prenijeti na drugi i tamo rjesiti problem. To je svojstvo portabilnosti.

Program pisan u VPJ se lakse pise, cita, odrzava i iz sintakse shvaca njegova funkcija, a program je i sam svoja dokumentacija.

Pisanje programa na VPJ je brze za pet i vise puta (ovisno o VPJ).

Dobro napravljeni VPJ po svojoj koncepciji onemogucuju nastajanje sintaktickih i semantickih gresaka raznih vrsta.

Programske jezike se moze podijeliti u grupe prema nekom kriteriju. Jedna od cescih klasifikacija je s obzirom na lakocu programiranja po kojoj se programi dijele na:

1. Nizi programski jezici
2. Visi programski jezici

Visi programski jezici dalje se mogu dijeliti u grupe (mada neki od njih imaju atribute vise grupa) prema nacinu postavke rijesenja problema na:

Visi programski jezici

1. Imperativni jezici
   1.1. Sekvencijalni jezici
   FORTRAN, COBOL, ALGOL, C, ADA, PL/1, NIT, SIMULA, PASCAL, SIMSCRIPT, PASCAL, BASIC ...

   1.2. Nesekvencijalni jezici (konkurentni)
   CONCURENT PASCAL, ADA, EDISON, MODULA CSP, OCCAM ...

   1.3. Simulacioni jezici (simuliranje kontinuiranih i diskretnih dogadjaja)
   DYNAMO, GPSS, SIMULA, SIMSCRIPT, CSMP, ...

2. Objektno orjentirani jezici
   SMALLTALK, ACTi, ADA ...

3. Aplikativni (funkcionalni) jezici
   LISP, FORTH, SNOBOL, ID, LUCID, VAL, VALID, FP, SASL, HOPE ...

4. Logicki jezici
   PROLOG, SETL ...

5. Prozorsko-orijentirani jezici
   VISICALC, MULTIPLAN, LOTUS 1-2-3, ...

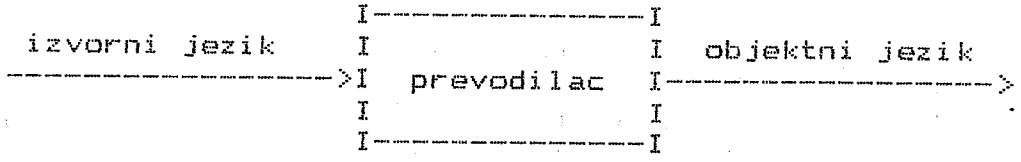
Pocetak razvoja programskih jezika zapocinje 1946. godine zajodno s razvojem modernih univerzalnih digitalnih racunarskih strojeva. U pocetku, do 1953. godine, su koristene razne metode za pisanje strojnih programa. John Backus je 1953. predlozio upravi IBM razvoj prvog viseg programskog jezika zvanog IBM Formula Translation System ili krace FORTRAN. Ideja je bila da se napise program prevodilac (kompajler) koji ce citati program koji je napisao programer sintaksom bliskom covjeku i prevoditi ga u strojni program blizak racunaru.

Neki znacajniji dogadjaji vezani za daljnji razvoj VPJ su:

1954 - Izradjena specifikacija za FORTRAN jezik
1957 - Zavrseno programiranje kompajlera za jezik FORTRAN I koji je trajao 18 covijek godina.
MART 1958 - Zavrsen prevodilac za FORTRAN II
KRAJ 1958 - Zavrsen prevodilac za FORTRAN III
1960 - Zavrsen prevodilac za jezik COBOL (Common Business Oriented Language) ciju je izradu inicirao 1959. godine DOD (US Department of Defence)
1962 - Zavrsen prevodilac za FORTRAN IV
1964 - Zavrsen NPL (New Programing Language) od strane IBM koji se danas naziva PL/1
1966 - Niklaus Wirth kreirao jezik ALGOL-W. John Backus je u izvjestaju 1958 predlozio jezik IAL (International Algorithmic Language) koji je kasnije promjenio ime u ALGOL-58, a za koga je 1960 nacinjena konacna verzija jezika ALGOL-60. Komitet za jezik je 1968. stvorio novi jezik ALGOL-68.

1966 - Definiran novi standard za FORTRAN, takozvani ANSI FORTRAN 66
1966 - J.O.DAHL definira jezik SIMULA (Simulation Language)
1966 - Niklaus Wirth kreira jezik EULER
1968 - N Writh zapoceo projektiranje jezika PASCAL ciji je prevodilac zavrsen 1970 (ETH, ZURICH)
1977 - Uvodi se novi standard za FORTRAN, ANSI FORTRAN 77
1980 - Grupacija DOD definira jezik ADA

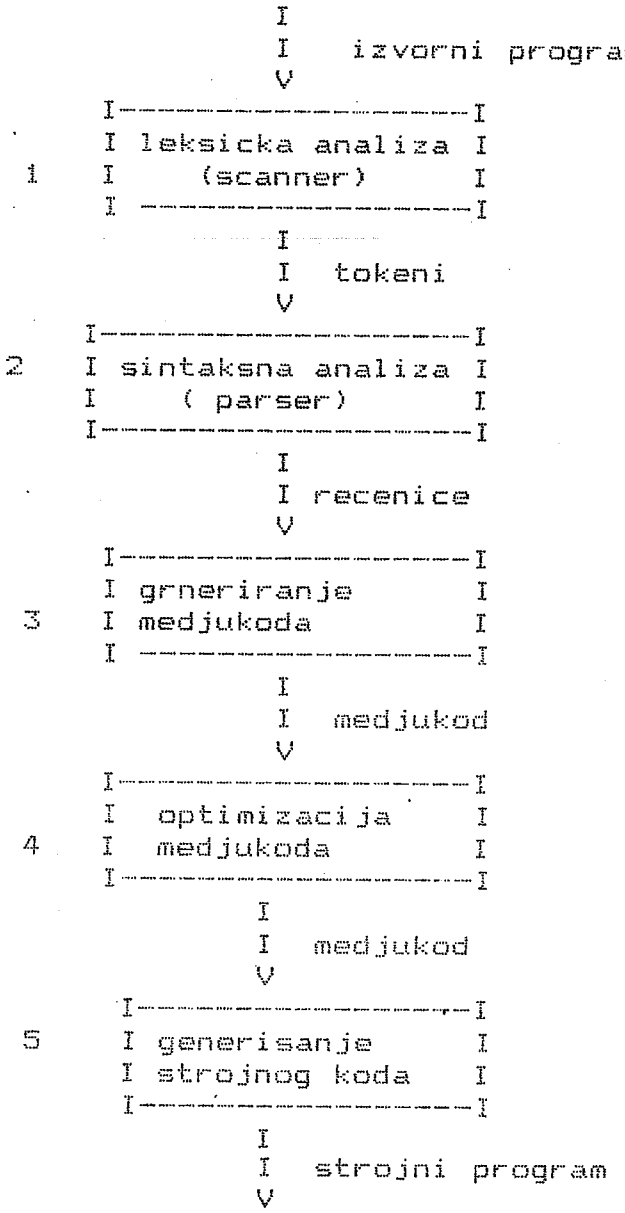
Napisan program izvan racunara potrebno je, programom za pisanje i obradu teksta (editor), unijeti u memoriju racunara. Tako upisan program naziva se izvorni (source language) program. FORTRAN prevodilac je program koji kao ulaz uzima neki program (source), koga je napisao covjek, i prevodi u drugi ekvivalentni program napisan na drugom programskom jeziku (object language, target language), koji razumije racunar. Taj program se naziva prevodilac (translator).



Sl. 1.2.

Prevodilac koji prevodi VPJ u strojni jezik naziva se kompajler (compiller, kompilator), a prevodilac koji prevodi VPJ u neki drugi VPJ naziva se predprocesor.

Kompajler kod prevodjenja obavlja niz funkcija, te se obzirom na njih graficki da prikazati (sl. 1.3.) opca struktura kompajlera.



Sl. 1.3.

Neki kompajleri ovih pet funkcija urade u jednom prolazu (veca operativna memorija racunara), tzv. "one pass" kompajleri, a do strojnog koda dodju u dva koraka, tako da u prvom prolazu urade prve tri funkcije a u drugom funkciju 4 i 5 tzv. "two pass" kompajleri.

Kod leksicke analize izvornog programa vrsi se rastavljanje izvornog teksta koji je u obliku niza znakova u niz rijeci (tokena) koje imaju neko logicko znacenje.

Sintaksni analizator (parser) dobiva kao ulaz niz rijeci (tokena) od leksickog analizatora i grupira ih u recenice (sintaksne cijeline).

Medjukod (intermediate code) je apstraktna notacija ne ovisna o detaljima racunara iz koje se lakse dolazi do strojnog programa, lakse se vrsi optimizacija te razdvaja sintaksna analiza od generisanja strojnog programa.

Cilj optimizacije je prevesti neki program u njemu ekvivalentan ali i efikasniji program, koji ce se brze izvrsavati.

Nakon prevodjenja izvornog programa (source) u objektni jezik (strojni kod), moze se visestruko izvrsiti (izvesti) program, uvijek ga inicirajuci jednom naredbom.

FORTRAN je kratica engleskih rijeci FORmula TRANslation (prevodilac formula). Namijenjen je za koristenje u znanosti za rjesavanje matematickih formula, ali se koristi i u racunarskoj grafici, statistici, tehnici, racunovodstvu itd. FORTRAN je standardiziran americkim ANSI propisima (American National Standard Programing Language FORTRAN). Prijenos programa moguc je na razne racunare, te je jedan od najtransportibilnijih programskih jezika. Mnogi racunari danas mogu raditi sa ovim jezikom (racunari: ICL1902A, VAX11, IBM1130, CYBER 70, IBM 360, SPERRY-UNIVAC 1100, svi IBM PC XT i IBM PC AT kompatibilci kao Olivetti PC M24, SPERRY PC,EPSON PC i dr.).

Tokom vremena FORTRAN se razvijao i do danas postoji vise nivoa jezika, razvijanih s idejom da svaki visi nivo ima sve mogucnosti nizeg i niz novih naredbi, koje programerima daju nove mogucnosti.

Nivoi FORTRAN-a:

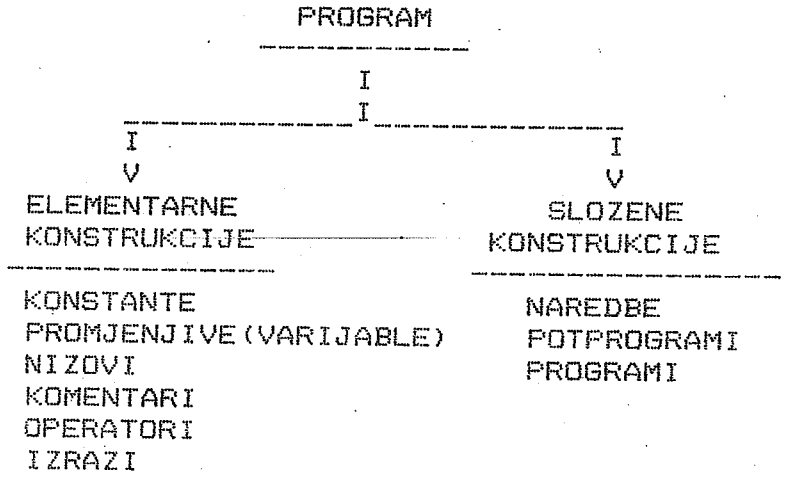
FORTRAN I
FORTRAN II
FORTRAN III
FORTRAN IV
FORTRAN V
FORTRAN 66
FORTRAN 77
ASCII FORTRAN (American Standard Code for Information Interchange)
FORTRAN 80

Cilj ovog teksta je da programere nauci programiranju standardnim FORTRAN 77. FORTRAN 77 je standardiziran od American National Standards Institute, Inc.(ANSI), u ANSI X3.9-1978. Standardizacija je povecala produktivnost programera, okupila i ujedinila niz dobrih ideja za podizanje nivoa mogucnosti jezika, samim tim smanjila troskove programiranja, povecala kvalitetu programa, omogucila njegovo lakse odrzavanje i modifikaciju te omogucila transportibilnost na razlicite racunare.

ASCII FORTRAN i FORTRAN 80 nisu standardizirani, pa funkcije koje su u njima ugradjene kao prednost, postaju mana u slucaju transportabilnosti (onemogucuju prenos).

Neka u tekstu navedena ogranicenja vezana za mogucnosti jezika ne ovise o standardu vec o konkretnom proizvodjacu racunara, dana su za racunare SPERRY-UNIVAC serije 1100. Slicna ogranicenja imaju i ostali konkretni prevodioci drugih proizvodjaca. Navodjenjem ovih vrijednosti citalac ima bolji uvid u domenu primjene sintakse jezika(npr. najveci i najmanji cijeli broj, brij redova nastavaka jedne naredbe i dr.).

Tipovi objekata, od kojih se sastoji program, s aspekta slozenoisti, dijele se na elementarne i slozene konstrukcije.



2. E L E M E N T A R N E K O N S T R U K C I J E

Ovaj dio prirucnika ima za cilj da programerima ukaze na mogucnosti elementarnih konstrukcija (ogranicenja, tipove, nacine rjesavanja), te sluzi kao podsjetnik i zahtjeva barem jedno detaljno citanje.

2.1. Simboli

Osnovna jedinica za gradnju jezika su simboli. FORTRAN jezik je sastavljen od:

Slova	A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z	
	(engleska abeceda, velika slova)	26 slova
Dekadne cifre	0,1,2,3,4,5,6,7,8,9	10 cifara
Specijalni znakovi iz ASCII skupa znakova		13 znakova
praznina	(prikazana kao "b" u ovom tekstu)	
=	jednako	
+	plus	
-	minus	
*	zvjezdica	
/	razlomacka crta	
(	otvorena zagrada	
)	zatvorena zagrada	
,	zarez	
.	tocka	
:	dvotocka	
'	apostrof	
$	dolar	

Slova, brojevi i specijalni znakovi zajedno se zovu karakteri (simboli, znakovi). Koji skup karaktera ce se koristiti ovisi o racunaru. Pored osnovnog skupa karaktera za gradnju jezika, programu je dostupan rad i sa ostalim iz skupa ASCII karaktera (vidi Dodatak A.). U ASCII skup karaktera su ukljucena i mala slova, koja mozemo koristiti u programu.

U programu se koriste razliciti objekti (elementi jezika) nastali kombinacijom karaktera. Neki od njih navedeni su kao primjer, a u daljnjem tekstu su detaljno objasnjeni.

A. Logicke konstante

.TRUE.	istina (1)
.FALSE.	neistina (0)

B. Aritmeticke operacije

+	zbrajanje
-	oduzimanje
*	mnozenje
/	dijeljenje
**	potenciranje

C. Znaci za operacije usporedjivanja

.LT.	<
.LE.	<=
.EQ.	=
.NE.	> < (razlicito)
.GT.	>
.GE.	>=

D. Logicke operacije

.NOT.	ne
.AND.	i
.OR.	ili
.EQV.	jednako
.NEQV.	razlicito

E. Sluzbene rijeci

DO
IF
FORMAT
READ
.
.
.

Ako neki od elemenata jezika napisemo malim slovom (sto nije greska) kompajler ce kod prevodjenja prevesti malo slovo u veliko.

2.2. KONSTANTE

Konstante su objekti u jeziku cija je vrijednost odredjena tipom konstante i inicijalnom vrijednoscu. Vrijednost konstante se ne moze mijenjati. Konstante se mogu podijeliti u kategorije:

2.2.1. Numericke konstante
2.2.2. Literalne (karakter) konstante
2.2.3. Logicke konstante

2.2.1. Numericke konstante

A. INTEGER konstante
B. REAL konstante
C. DOUBLE PRECISION REAL konstante
D. COMPLEX konstante

A. INTEGER konstante (cijeli brojevi)

INTEGER konstante su cijeli brojevi (negativni cijeli brojevi, nula, pozitivni cijeli brojevi) sa istim znacenjem i zapisivanjem kao u matematici.

npr.: 12, 0, +10325, -17, (+3=3), itd.

Apsolutna vrijednost INTEGER konstante moze biti manja ili jednaka od broja:

$$2^{35}-1=34\ 359\ 738\ 367$$

$$-3{,}4 * 10^{10} <= \text{kons.} <= 3{,}4 * 10^{10}$$

U daljnjem tekstu definirane su ekstremne vrijednosti za racunar SPERRY UNIVAC serije 1100. Ekstremne vrijednosti su ovisne o vrsti racunara.

B. REAL konstante (realni brojevi)

Postoje dva nacina zapisa realnog broja kao real konstante, matematicki zapis realne konstante i eksponencijalni oblik realne konstante. Decimalne brojeve

5,16 ; -863,175 ; ,0 ; 0,0 ; 0, ; 19,0 ; 0,000000001

pisemo u obliku realne konstante unutar programa kao :

5.16 ; -836.175 ; .0 ; 0.0 ; 0. ; 19. ; .000000001,

izmjenom decimalnog zareza "," u decimalnu tocku ".", koju je obavezno pisati, da bi se razlikovala REAL od INTEGER konstante. Konstanta moze imati od 1 do 9 decimalnih mjesta.

Eksponencijalni oblik real konstante je nacin zapisa konstante kao produkta realne konstante i potencije s bazom 10. Opci oblik: aEpe ;gdje je: a - normalna realna konstanta, E - oznaka eksponencijalnog oblika realne konstante, p - predznak (+ ili -), e - eksponent broja 10. Predznak + nije obavezno pisati.

$$\text{npr.: } 2,57*10^{2} = 2.57E+02$$

$$0,00000237 = 2,37*10^{-7} = .0237E-05$$

Dozvoljene ekstremne vrijednosti REAL konstanti

$$+/-10^{-38} <= \text{kons.} <= +/-10^{38}$$

npr.: Dobro napisane REAL konstante su:
0.0 ; .0 ; 7. ; -19.87 ; 3.0E2 (umjesto 300.0) ; 3E2 (umjesto 300.0) ; 8.0E+3 (umjesto 8000.0) ; 8.0E-2 (umjesto 0.08)

C. DOUBLE PRECISION REAL konstante

Ukoliko vrijednosti konstante prelaze dozvoljene ekstremne vrijednosti realne konstante ili zahjteva vise od 9 decimalnih mjesta, koristi se konstanta s dvostrukom preciznoscu. Konstantu s dvostrukom preciznoscu oznacava se umetanjem slova "D" na mjesto slova "E" kod realne konstante. Ova konstanta moze imati od 1 do 18 decimalnih mjesta (18 cifri), a apsolutna vrijednost eksponenta moze biti maksimalno 308

npr.: 57,0=0.57D+02

$$1,23*10^{39} = 1.23D+39$$

$$1,45*10^{-308} = 1.45D-308$$

Dozvoljene ekstremne vrijednosti DOUBLE PRECISION REAL konstanti:

$$+/-10^{-308} <= \text{kons.} <= +/-10^{+308}$$

npr.: Dobro napisane realne konstante s dvostrukom preciznoscu su: 0.0D0 (za 0) ; 1.0D0 ili 1D0 (za 1) ; 18.7D+1 ; -35.76D-18 ; 1345.0D+15 ; 123.4567891D0 ; .1234567891D3 (vrijednost zadnje dvije konstante je jednaka)

D. COMPLEX konstante

Kopleksni broj Z po definiciji je suma realnog i imaginarnog dijela Z=R+iI. U programu, kompleksni se broj pise u obliku uredjenog para Z=(R,I) gdje je: R-realni dio, a I-imaginarni dio. Oba clana moraju biti ili INTEGER ili REAL. Ukoliko je R=0 ili I=0 obavezno je navesti oba clana.

```
npr.: (2.3,8.31)          =   2,3+8,31i
      (5,-3)              =   5-3i
      (2.45,128.E-02)     =   2,45+1.28i
      (0.245E+01,1.28)    =   2.45+1,28i
      (0,+7)              =   7i
      (0.0,0.0)           =   0,0+0,0i
```

2.2.2. Literalne konstante

Osim brojeva i logickih konstanti moguce je bilo koju kombinaciju ASCII karaktera, koji stoje na raspolaganju jeziku, proglasiti literal konstantom, do makimalno 511 karaktera. Kombinaciju karaktera, koju se zeli proglasiti literal konstantom u programu, umece se izmedju znakova apostrofa. Ako je apostrof dio teksta, pisu se dva uzastopna apostrofa.

```
npr.:
      'TEXT'
      'AB1$*'
      'RIJEKA JE PRIMORSKI GRAD',
      'A, B  C'
      'THAT''S'
```

2.2.3. Logicke konstante

Logicka konstanta,kao u algebri logike, moze poprimiti dvije vrijednosti.

.TRUE.	istinito, tocno, u algebri logike 1
.FALSE.	lazno, netocno, u algebri logike 0

2.3. Varijable

Varijable su velicine u programu koje mogu poprimiti razlicite vrijednosti u toku odvijanja programa, a u matematici odgovaraju opcim brojevima a, b, c, x, y, z, ... Varijabla se sastoji iz imena i pripadne vrijednosti. Ime moze imati od 1 do 6 znakova ,i prvi znak mora biti slovo. U imenu varijable mogu se pojaviti cifre (0,1,2,3,4,5,6,7,8,9). Programer moze izabrati proizvoljno ime. Upotreba sluzbenih rijeci GO TO, READ, FORMAT itd., je zabranjena za imenovanje varijabli. Varijabla je simbolicko ime koje definira memorijsko mjesto u kome se nalazi tekuca vrijednost. Broj memorijskih mjesta i tip vrijednosti koji je postavljen u to mjesto je odredjen deklariranjem varijabli (implicitno ili eksplicitno). Dozvoljeni tipovi varijabli su: INTEGER, jednostruko ili dvostruiko precizni REAL, jednostruko precizni COMPLEX, CHARACTER, LOGICAL. Varijable se dijele u tipove (vrste) na isti nacin kao konstante.

Tipovi varijabli:

2.3.1. Numericke varijable
2.3.2. Literalne (karakter) varijable
2.3.3. Logicke varijabvle

2.3.1. Numericke varijable

A. INTEGER varijable
B. REAL varijable
C. DOUBLE PRECISION varijable
D. COMPLEX varijable

U matematici varijablama: P, a, b, V, itd. iz formula:

$$P=\frac{a * Va}{2} \quad \text{ili} \quad A=B*C$$

odgovaraju u FORTRAN-u imena, npr.: AMORTI, IVO, I3, X111, a1B1, itd.

A. INTEGER varijable (cijelobrojne varijable)

INTEGER varijabla moze poprimiti vrijednost INTEGER konstante. Implicitno se podrazumjeva da je varijabla tipa INTEGER ako ime varijable pocinje sa slovom:I,J,K,L,M ili N. Ovo je tradicionalni nacin odredjivanja tipa INTEGER varijabli.

npr.: I1
IVO
MO
MMN

Ako se zeli u programu imati varijablu sa pocetnim slovom razlicitim od I,J,K,L,M,N (kao: CIJENA, AKON, B, ZZ, ...) koristi se naredba INTEGER, te je tip varijable INTEGER bez obzira na prvo slovo. Naredba INTEGER sluzi za eksplicitno deklariranje tipa varijable.

npr.:

```
INTEGER CIJENA,AKON,B,Z,O...
```

B. REAL varijable

Implicitno se podrazumjeva da je varijabla tipa REAL ,ako varijable pocinju sa slovom:
A,B,.....G,H,O,P,....X,Y,Z
npr.:GORE, PUT, BRZINA, W12, ZOS, A123, X1X1 ...
Ako se zeli u programu imati REAL varijabilu sa pocetnim slovom /I,J,K,L,M,N/, koristimo naredbu REAL (eksplicitno deklariramo).

npr.:

```
REAL IVO,MASA,GRA,..
```

U FORTRAN-u se implicitno moze odrediti tip varijable (INTEGER ili REAL), te zbog toga on nije strogo tipski jezik, sto jesu novije razvijeni programski jezici. Ova mogucnost moze izazvati gresku; bilo da smo zeljeli imati REAL varijablu a imamo implicitno INTEGER ili obrnuto; koju prevodilac ne moze otktiti. Program sa ovakvom greskom se izvrsava i daje rezultate , naoko tacne, ali u stvari pogresne. Ova greska se tesko otkriva pa se preporucuje: EKSPLICITNO DEKLARIRATI TIPOVE SVIH VARIJABLI KOJE SE KORISTE U PROGRAMU.

C. DOUBLE PRECISION REAL varijable

Ako se u programu zeli imati neku realnu varijablu s dvostrukom preciznoscu, obavezno se mora naredbom DOUBLE PRECISION specifirati tip varijable.
npr.:

DOUBLE PRECISION ABR,Z1,IZNOS,PUT,KR11

D. COMPLEX varijable

Sve kompleks varijable je obavezno eksplicitno deklerirati naredbom COMPLEX. To su sve varijable u programu koje ce imati vrijednost kompleks konstante.
npr.:

COMPLEX B,Z1,IZ3

2.3.2. Literalne varijable

Ako se zeli imati u programu varijabla koja ce poprimiti vrijednost razlicitih literal konstanti, obavezno se mora deklarirati (najaviti) na pocetku programa naredbom CHARACTER ciji je opci oblik:

CHARACTER*n A,B*m,D11 gdje je:

n - duzina karakter varijable u znakovima
m - nova duzina karakter varijable u znakovima
A,B,D11- varijable tipa karakter

npr.:

CHARACTER*10 A,B,C1*5,C2*1,GRAD*25

Varijabla A i B su duzine najvise 10 karaktera, varijabla C1 je duzine 5 karaktera, C2 je duzine 1 karakter, a varijabla GRAD je duzine 25 karaktera.

2.3.3. Logicke varijable

Sve logicke varijable treba eksplicitno deklarirati naredbom LOGICAL. To su varijable u programu koje ce imati vrijednost logicke konstante.
npr.:

LOGICAL A1,IME,XO

2.4. Implicitno odredjivanje tipa varijable

Ukoliko se zeli izmjeniti implicitno deklariranje tipova varijable i odrediti varijable s pocetnim slovima I,J,K,L,M,N tipa REAL, a varijabile koje pocinju s ostalim slovima tipa INTEGER, te ako se zeli implicitno deklarirati varijable tipa LOGICAL, CHARACTER, DOUBLE PRECISION ili COMPLEX, koristi se naredbu IMPLICIT ciji je opci oblik:

IMPLICIT tip (sl1-sl2),tip(sl3,sl4...),...
gdje je:

tip-jedan od clanova skupa (INTEGER, REAL, LOGICAL, COMPLEX, DOUBLE PRECISION)
sl1,sl2,...-slova engleske abeced

```
npr.:  IMPLICIT INTEGER (A-H),REAL(I-K)
       IMPLICIT LOGICAL (L),CHARACTER*10(B,F-K)
```

U drugom primjeru sve varijabile koje pocinju sa slovom L su logicke, a sve koje pocinju sa slovom B i sa slovom F,G,H,I,J,K su karakter duljine 10 znakova. Ostala slova su po prethodnoj konvenciji. Pri tom treba paziti da se isto pocetno slovo ne dodijeli razlicitim tipovima varijabli.

ZADACI:

1. Napisati brojeve kao realne konstante

a)	10	–	10. ili 1.E1
b)	3,324	–	3.324
c)	0,00008	–	0.8E-4
d)	-124,7	–	-1.247E2
e)	15	–	1.5E+01
f)	0.28	–	2.8E-1
g)	-18.23	–	-1.8E1
h)	5624.023	–	5.624023E3
i)	0.003	–	3.E-03

2.5. Matrice

Sistem od m n brojeva, rasporedjenih u pravokutnoj tablici od m redaka i n stupaca nazivamo matrica (dvodimenzionalna matrica). Matrica je skup varijabli koga nazovemo jednim simbolickim imenom. Svaku varijablu posebno nazivamo clanom matrice. Kao i varijable, matrice mogu biti: cjelobrojne, jednostruko ili dvostruko precizno realne, kompleksne, literalne ili logicke.

oznaka:

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \quad I=A(m,n)$$

m – maksimalni broj redova (vrsta)
n – maksimalni broj stupaca (kolona)

Svaki pojedini clan matrice zauzima tacno jednu poziciju, odredjenog reda i stupca . Clan matrice (2,1) lezi u drugom redu: prvom stupcu itd.. Opca oznaka za oznacavanje svakog pojedinog clana matrice jest A(i,j), gdje je i-broj reda j-broj stupca u kome se nalazi promatrani clan matrice. Clan A(2,5) je znaci clan matrice A na presjecistu drugog reda i petog stupca.

Jednodimenzionalnom matricom (nizom) naziva se skup od n brojeva, kod kojih se tocno zna koji je prvi, koji je drugi itd., do posljednjeg.
oznaka:

```
    --                          --   --                 --
B=Ia     a      .....     a    I= Ia   a    .....   a   I=B(n)
  I 11   12              1n I  I 1  2             n I
    --                          --   --                 --
```

Mogu se definirati maksimalno 7-dimenzionalne matrice A(n,m,i,j,k,l,o). Primjetimo da vec trodimenzionalnu matricu ne mozemo prikazati u obliku pravokutne tablice. Vec je to niz pravokutnih matrica. Ako je dana dvodimenzionalna matrica,

```
           --        --
           I 7     5  I
           I          I        m=3 - tri reda
A(3,2)=    I 3    -1  I        n=2 - dva stupca
           I          I
           I 8     9  I
           --        --
```

tada je: A(1,1)=7, A(1,2)=5, A(2,1)=3, A(2,2)=-1, A(3,1)=8, A(3,2)=9. Kako bi se razlikovale varijable koje primaju samo jednu vrijednost od matrica koje primaju vrijednosti jednodimenzionalne , dvodimenzionalne, itd. tablice, eksplicitno se deklarira matrice naredbom DIMENSION ciji je opci oblik:

```
DIMENSION  A(n),B(m,k),C(x1,x2,x3,x4,x5,x6,x7),...
```

Gdje je:
A,B....- ime matrice(varijable)
n- broj clanova matrice A (cijeli broj)
m- broj redova matrice B (cijeli broj)
k- broj stupca matrice B (cijeli broj)

npr.:

```
DIMENSION ARR(7,11),C(5),K(126,115,37)
```

Dimenzija matrice odredjena je implicitno, u zagradi iza imena matrice, brojem upisanih koeficijenata odvojenih zarezom. Broj clanova matrice ne smije biti veci od parametra koji definira velicinu dimenzije matrice. Naredbama za eksplicitno deklariranje tipa varijable (INTEGER , REAL, LOGICAL, COMPLEX, DOUBLE PRECISION, CHARACTER i naredbom COMMON samo u potprogramu), mogu se deklarirati matrice i nije potrebna naredba DIMENSION.

npr.:

```
DOUBLE PRECISION  ARR(15),A11(7,11,3)
REAL B(3,2),C(8)
DIMENSION FF1(5,7,6)
```

U zadnjem primjeru definirana je trodimenzionalna matrica FF1 dimenzija 5x7x6.

Matrica moze biti bilo kojeg tipa, kao i varijabla i za imenovanje vrijedi isto sto i za tipove varijabli. Matrica moze biti numericka, logicka ili literal. Matrica moze imati maksimalno 263143 clana.

npr.:

```
DIMENSION A(263143),B(263,1000)
```

Da se dodje do pojedinog clana matrice (npr. dvodimenzionalne) u programu je potrebno specificirati redak i stupac u kome se nalazi trazeni clan. Ne moze se oprerirati istovremeno sa citavom matricom vec sa svakim clanom pojedinacno, pomocu indeksa broja reda i stupca A(i,j), pri cemu je indeks bilo koja linearna kombinacija

$$C_1 X_1 + C_2 X_2 + \ldots + C_n X_n$$

gdje su $C_1, \ldots, C_n$ INTEGER konstante a

$X_1, \ldots, X_n$ INTEGER varijable.

npr.: A(1) - prvi clan jednodimenzionalne matrice A.

A(I) - I-ti clan matrica A. Koja ce vrijednost niza biti koristena, ovisi o vrijednosti I

npr: Ako su dani clanovi matrice XA i UV

XA(k-3) ako je k=16, onda je to trinaesti clan matrice XA

UV(2*I,3*J-1), ako je trenutna vrijednost varijable I=6, a J=3, onda je UV(12,8) clan matrica UV iz 12. reda i 8. stupca.

UV(3/3,SQRT(4),4) je clan matrice u prvom redu, drugoj koloni i cetvrtoj u nizu dvodimenzionalnih tablica.

Vrijednost indeksa varijable ne smije biti manja od jedan i veca od dimenzije matrice iz naredbe DIMENSION.

Trodimenzionalnu matricu A(3,3,2) racunar memorira clan po clan redosljedom:

A(1, 1, 1), A(2, 1, 1), A(3, 1, 1), A(1, 2, 1), A(2, 2, 1),
A(3, 2, 1), A(1, 3, 1), A(2, 3, 1), A(3, 3, 1), A(1, 1, 2),
A(2, 1, 2), A(3, 1, 2), A(1, 2, 2), A(2, 2, 2), A(3, 2, 2),
A(1, 3, 2), A(2, 3, 2), A(3, 3, 2).

Uocimo da je redoslijed memoriranja clanova takav, da se najprije mijenja prvi indeks, pa drigi indeks i tako do posljednjeg.

Matrice se mogu definirati samo u potprogramima, ali ne COMMON naredbom, i na nacin:

opci oblik:

```
DIMENSION A(d1:d2,e1:e2, ... ,g1:g2), ...
```

gdje je:

A - ime matrice

d1,e1,g1 - aritmeticki izraz sastavljen od cijelobrojnih konstanti i cijelobrojnih varijabli, koja odredjuje najnizi clan matrice. On moze biti negativan broj ili nula. Ako nije naveden smatra se da je jednak 1, odnosno da je prvi clan matrice A(1).

d2,e2,g2 - aritmeticki izraz sastavljen od cijelobrojnih konstanti i cijelobrojnih varijabli, koja odredjuje poslijednji clan matrice A. Ova vrijednost moze biti (*), te se ne mora brinuti o maksimalnoj dimenziji matrice. Vrijednosti parametra d2,e2,g2 moraju biti vece ili jednake od parametara d1,e1,g1.

npr:

Dobro definirane matrice su:

```
DIMENSION MAT(-12:+12), A(1:4*m,1:100)
COMPLEX TABLI(0:10,0:20,0)
```

Matrica MAT je jednodimenzionalna sa 25 clanova, ciji je prvi clan MAT(-12) a dvadesetpeti clan MAT(12). Matrica A je dvodimenzionalna, ciji je prvi clan prve dimenzije jednak vrijednosti 1, a poslijednji clan prve dimenzije je 4*m.

2.6. Izrazi

Izraz je skup varijabli, konstanti , clanova matrica i operatora.

Tipovi fortranskih izraza

2.6.1. Aritmeticki izrazi
2.6.2. Literal izrazi
2.6.3. Logicki i relacijski izrazi

2.6.1. Aritmeticki izrazi

Aritmeticki izraz se sastoji od aritmetickih konstanti, varijabili, koje su tipa INTEGER, REAL (jednostruke ili dvostruke preciznosti) ili COMLEX, standarnih matricnih funkcija, zagrada i aritmetickih operatora. Vrijednost aritmetickog izraza je uvijek: INTEGER, REAL (jednostruke ili dvostruke preciznosti), ili COMPLEX.

Aritmeticki operatori

Iprioritet I	Operator I	naziv operatora I	primjer aritmetickog izraza I
5	+	zbrajanje	7+3
5	-	oduzimanje	-5.1+8.3
4	*	mnozenje	a*1.23E-03
4	/	dijeljenje	8/B
3	**	potenciranje	B**3
funkcije i zagrade imaju veci prioritet			
2	sin()	funkcije	sin(3.14)
1	()	zagrade	((A+B)**(1./17))= $\sqrt[17]{a+b}$

Konvencije u koristenju aritmetickih izraza:

1.Operatori "+" i "-" se upotrebljavaju i kao predznak. Predznak ne moze stajati pored aritmetickih operatora ; nije dobro A * -B, vec treba A * (-B). Dva aritmeticka operatora ne smiju stajti jedan pored drugoga izuzev ** sto je operator potenciranja.

2. Izrazi se rjesavaju s lijeva na desno po prioritetu operatora npr.: Najprije potenciranje, mnozenje pa oduzimanje.

```
A * D - C ** 2
-----   ------
  2       1

---------
    3
```

3. Rezultat aritmetickog izraza ovisi o tipu operanda. Rezultat je onoga tipa kojeg je tipa operand najviseg ranga. Tipovi operanda su po rangu od najviseg do najnizeg slozeni redom: kompleksni, realni s dvostrukom preciznoscu, realni, cjelobrojni.

4. Aritmeticke operatore je obavezno pisati. Za produkt brojeva A i B nije dobro pisati AB sto je dovoljno u matematici, vec treba biti A*B. Aritmeticki operatori se ne podrazumjevaju i treba ih pisati.

5. Zagrade daju najveci prioritet operacijama unutar njih. Ukoliko ima vise zagrada od potrebnog minimuma rezultat racunanja ce biti ispravan. U nekoliko primjera koji su napisani bez zagrada, prikazan je redosljed izvrsavanja upotrebom zagrada.

ako pise		kao da pise
A - B * C * D	=	(A - ((B * C) * D)
A ** (B - C) * D	=	((A ** (B - C)) * D)
- A ** B	=	- (A ** B)
-A*B**C	=	((-A)*(B**C))
A**B**C	=	A**(B**C)

U zadnjem primjeru ne vrijedi pravilo izvrsavanja operacija po prioritetu s lijeva na desno , sto je i jedina iznimka, gdje se prvo izvrsava B**C. Potrebno je pisati izraz sa zagradama

$$(A^{**}B)^{**}C = (A^{B})^{C}$$

ukoliko se zeli prvo potenciranje A**B.

6. Kod dijeljenja cjelobrojnih operanada rezultat je cijeli broj, sto znaci da se ostatak dijeljenja odbacuje.

$$\frac{I}{N} \quad ===> \text{ rezultat je cijeli broj}$$

Ako je I=8, N=3 tada je rezultat=2. To nije zaokruzivanje na prvi blizi, vec na prvi nizi cijeli broj, jer se decimale odbacuju.

Zadaci:

1.Zadane matematicke formule napisati u obliku aritmetickih izraza.

$$ab+cd^{z} \quad ; \quad 3x^{2}+2x^{2}+1 \quad ; \quad \frac{ab}{c^{2}+x^{2a}} \quad ; \quad \sqrt{2gh} \quad ;$$

$$\sqrt[13]{I^{15}_{a}} \quad ; \quad (-A)^{B^{C}} \quad ; \quad ((-a)^{2}+b\frac{d}{a})^{2} \quad ;$$

$$\frac{A^{2}}{(B+C)} - \frac{4}{D} + 3$$

2.6.2. Literal izrazi

Literal izraz je skup literal konstanti, varijabili, clanova matrica i literal operatora.

Opci izraz:

$$e_{m} \,//\, e_{n}$$

gdje je:

// - literal operator udruzivanja

e_m - literal(konstanta, varijabila, clan niza) duzine m

e_n - literal(konstanta, varijabile, clan niza) duzine n

Rezultat udruzivanja je literal izraz duzine m+n

npr.:

'FORTRAN '//'TECAJ'==>FORTRAN TECAJ

'"3.MAJ"'//' SOUR BI'==>"3.MAJ" SOUR BI

'TYPE'//'123'==>TYPE123

'RAT'//' I '//'MIR'=>RAT I MIR

Ako A(5,3) je '50 DIN' i B(1) je 'SUMA' tada rezultat izraza

B(A)//'='//A(5,3) je SUMA=50 DIN

Zadatak: Napisati literal izraz koji bi dao rezultat

"ZIVIO 1. MAJ" ako A(1) je 'ZIVIO', B(2) je '1.', C(3) je 'MAJ'

2.6.3. Logicki i relacijski izrazi

Opci oblik: e operator e
1 2

gdje je: e i e - logicki ili aritmeticki operandi
1 2
operator - logicki ili aritmeticki operator

Ovi izrazi se dijele u dvije grupe: 1. Cisti logicki izraz i 2. Relacijski izrazi. Za obje grupe vrijedi, da je rezultat izraza uvijek .TRUE. ili. FALSE.. U cistom logickom izrazu operandi su samo logicki, a u relacijskom operandi mogu biti : logicki i relacijski irazi. Relacijski operatori imaju veci prioritet(manji broj) i prvi se rjesavaju u izrazu. Zagrade(kao kod aritmetickih izraza) imaju najveci prioritet.

prio-ritet	operator	znacenje operatora
	Relacijski operatori	
1	.GT.	vece od
1	.GE.	vece ili jednako
1	.LT.	manje od
1	.LE.	manje ili jednako
1	.EQ.	jednako
1	.NE.	razlicito
	Logicki operatori	
2	.NOT.	ne
3	.AND.	i
4	.OR.	ili
5	.EQV.	jednako
5	.NEQV.	razlicito

Relacijski operatori se jos nazivaju operatori poredjenja jer stoje izmedju dva aritmeticka ili literal izraza koje uporedjujemo. Aritmeticki izraz tipa COMPLEX je dozvoljen samo za relacijske operatore .EQ. i .NE.. Ako je jedan operator tipa CHARACTER i drugi mora biti istog tipa. Matematicka relacija poredjenja I < A+7 odgovara relacijskom izrazu I.LT.(A+7) koji moze biti istinit ili lazan ovisno o vrijednostima varijabli I i A u programu. Rjesavanje istinosti izraza u kojem ima logickih operatora, jednako je postupku iz matematicke logike. Rezultat logickog izraza ovisi i o vrijednostima logickih operanda i o logickom operatoru. Nize su dane tablice istinitosti logickog izraza (suda), za svaki operator posebno, sa svim kombinacijama vrijednosti operanda.

Negacija(.NOT.) za izraz .NOT.A

A	.NOT.A
.TRUE.	.FALSE.
.FALSE.	.TRUE.

Izraz ima suprotnu vrijednost od operanda A.

Konjukcija (.AND.) za izraz A.AND.B

A	B	A.AND.B
.TRUE.	.TRUE.	.TRUE.
.TRUE.	.FALSE.	.FALSE.
.FALSE.	.TRUE.	.FALSE.
.FALSE.	.FALSE.	.FALSE.

Ako su oba operanda A i B istinita izraz je istinit, u protivnom izraz je lazan

Disjunkcija (.OR.) za izraz A.OR.B

A	B	A.OR.B
.TRUE.	.TRUE.	.TRUE.
.TRUE.	.FALSE.	.TRUE.
.FALSE.	.TRUE.	.TRUE.
.FALSE.	.FALSE.	.FALSE.

Ako su oba operanda A i B lazna izraz je lazan, u protivno izraz je istinit

Ekvivalencija (.EQV.) za izraz A.EQV.B

A	B	A.EQV.B
.TRUE.	.TRUE.	.TRUE.
.TRUE.	.FALSE.	.FALSE.
.FALSE.	.TRUE.	.FALSE.
.FALSE.	.FALSE.	.TRUE.

Ako su oba operanda A i B istinita ili oba lana izraz je istinit, u protivnom izraz je lazan

Negacija ekvivalencije (.NEQV.) za izraz A.NEQV.B

A	B	A.NEQV.B
.TRUE.	.TRUE.	.FALSE.
.TRUE.	.FALSE.	.TRUE.
.FALSE.	.TRUE.	.TRUE.
.FALSE.	.FALSE.	.FALSE.

Ako je vrijednost operanda A i B suprotna izraz je istinit, a u protivnom izraz je lazan

npr: Ako A je .TRUE. i B je .FALSE., onda je rezultat izraza

a) A.AND.B ==>.FALSE.

b) .NOT.B ==>.TRUE.

c) .NOT.(A.OR.B.) ==>F

T

d) 3.LT.15 ==> T

e) .NOT.10.LE.7.AND.A ==> T

f) .NOT.3.LE.4OR.B ==>F

g) A.NEQV.B ==>T

h) 'TIP'.EQ.'TOP' ==>F

i) A.GT.3.14 ==>T ako je vrijednost varijable A veca od 3.14

Zadatak: Naci istinitost izraza ako vrijednost logickih varijabli A je .TRUE., B je .FALSE. i cjelobrojnih varijabli I je 7, J je 10

a) .NOT.B.AND.NOT.A

b) .NOT.B.AND.I.LT.J

c) A.OR.NOT.10.NE.J

2.7. Hijerarhija operatora

Operatori(aritmeticki, literal, logicki i relacijski), funkcije i zagrade imaju razlicite medjusobne prioritete u FORTRAN izrazima pa je u slijedecoj tablici dat njihov medjusobni prioritet za razrjesavanje izraza. Veci broj oznacava manji prioritet. Zagrade imaju najveci prioritet sto znaci da se rjesavaju najprije izrazi u zagradama.

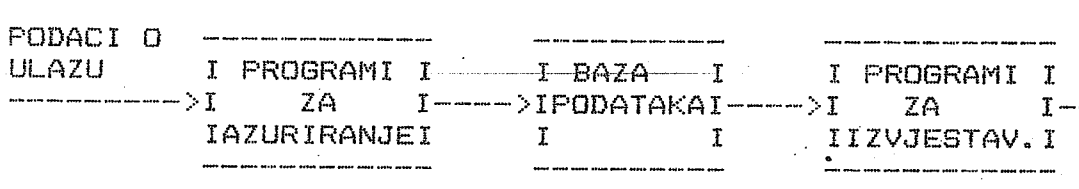
rang	vrsta
1.	zagrade (sve)
2.	funkcije (sve)
	aritmeticki operatori
3.	potenciranje(**)
4.	mnozenje i dijeljenje(* i /)
5.	zbrajanje i oduzimanje(+ i -)
	operator udruzivanja
6.	udruzivanje (//)
	relacijski operatori
7.	poredjenje (.GT.,.GE.,.LT.,.EG.,.NE.)
	logicki operatori
8.	ne (.NOT.)
9.	i (.AND.)
10.	ili (.OR.)
11.	jednako (.EQV.)
12.	razlicito (.NEQV.)

3. P R O G R A M

Programski jezik moze biti primjenjen za rjesavanje jednostavnijih problema (naci sumu sto clanova danog reda) i slozenih (izracunati brzinu broda u ovisnosti o snazi motora, formi brodskog trupa, masi broda i drugih parametara koji utjecu na brzinu). Kod jednostavnijih primjena pristupa se izradi modela programa, upotrebom simbola za crtanje dijagrama toka programa, i na osnovu njih kodira program za izvrsenje. Kod primjene racunara na slozene realne sisteme, cilj je projektirati informacioni sistem (IS), koji ce biti model realnog sistema i iz koga cemo dobiti potrebne informacije o realnom sistemu. Postupak projektiranja IS sastoji se od faza (Lazarevic 1979):

1. Analiza potreba i ekonomicnosti
2. Analiza zahtjeva korisnika
3. Logicko projektiranje
4. Programiranje
5. Instalacija i testiranje
6. Odrzavanje i modifikacija

Opcenitu arhiktekturu IS mozemo prikazati blok-dijagramom:



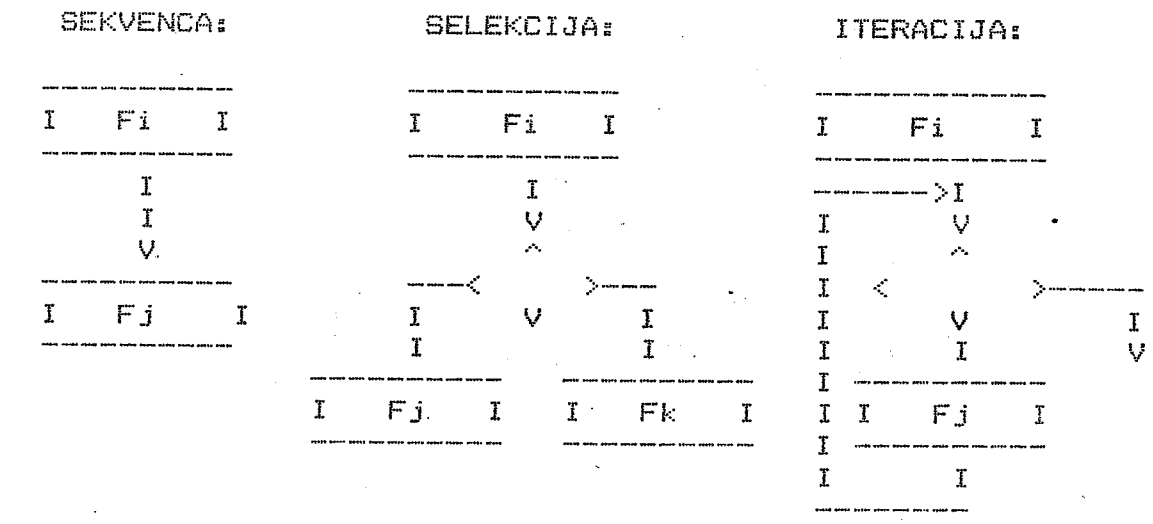
Iz ovkve arhitekture IS, program se definira kao uredjeni skup instrukcija koji transformise neki ulazni skup podataka u neki izlazni skup podataka (Chand, Yadav, 1980). Uredjeni skup instrukcija predstavlja niz funkcija koje program obavlja tokom svog izvrsenja. Svaka od funkcija je niz instrukcija. Funkcije su medjusobno povezane u logicku strukturu. Funkcije programa mogu biti: citanje ulaznih podataka, racunanje, kontrola, izvjestavanje, itd. . Strukturno programiranje (Dahl, Dijkstra, Hore, 1972) je niz ogranicenja u povezivanju osnovnih funkcija programa u logicku strukturu programa. Strukturno pisani programi su produktivniji, pouzdaniji, lakse se pisu i testiraju i imaju niz drugih prednosti. Program pisan strukturno podrazumjeva povezivanje osnovnih funkcija u tri pravilne strukture koje nazivamo: sekvencija, selekcija i iteracija (linijska, razgranata i ciklicka struktura).

Struktura tipa sekvencije povezuje funkcije Fi i Fj tako da nakon jednog izvrsenja funkcije Fi slijedi jedno izvrsenje funkcije Fj.

Struktura tipa selekcije povezuje funkcije Fi , Fj i Fk tako da nakon jednog izvrsenja funkcije Fi (i u ovisnosti o tom izvrsenju) slijedi ili jedno izvrsenje funkcije Fj ili jedno izvrsenje funkcije Fk.

Struktura tipa iteracije povezuje funkcije Fi i Fj tako da nakon jednog izvrsenja funkcije Fi slijedi vise izvrsenja funkcije Fj.

Dobre strukture se graficki daju prikazati blok-dijagramima:



Da bi program bio strukturan, pored povezivanja funkcija u dobre strukture, potrebno je skup instrukcija koje cine pojedine funkcije povezati u dobre strukture. Svaki se program moze realizirati strukturama tipa: sekvencija, selekcija i iteracija. Uocimo da je za pojektiranje IS, posao programiranja i testiranja programa samo jedna od niza aktivnosti, te se za siru primjenu programskih jezika preporucuje upoznavanje sa literaturom iz podrucja projektiranja IS.

3.1. Nacin pisanja programa

Naredjenja racunaru daju se FORTRAN programom napisanim FORTRAN jezikom, koji mora biti u skladu sa pravilima o pisanju programa.

Obrazac za pisanje programa

obiljezje		naredba	identifikacija
1 2 3 4 5	6	7 ... 72	73 ... 80

Obrazac za pisanje programa ima 80 kolona. U jednom redu upisuje se jedna naredba racunaru.

Od 1.do 5.kolone moze se upisati broj koji predstavlja obiljezje(adresu, labelu) naredbe. Svaka naredba moze imati labelu, ali produzetak naredbe u novi red ne smije imati labelu. Dvije naredbe ne smiju imati istu labelu.

U 6. kolonu, kada je naredba duza od 66 karaktera, upisujemo znak razlicit od "0" i praznine("b") i nastavljamo je pisati u novi red. Naredbu se moze prosiriti do 20 redova odnosno i vise ali do najvise 1320 znakova ne racunajuci "b". Maksimalna duzina naredbe ovisi o racunaru.

Od 7. do 72. kolone upisuje se naredba. Naredba moze biti zapoceta bilo gdje u tom intervalu.

Od 73. do 80. kolone moze se upisati bilo koji tekst, npr. za prepoznavanja programa. Kod prevodjenja u masinski kod ovaj dio teksta se ne uzima u obzir.

Komentar i objasnjenja za tumacenje programa slobodnim opisom pocinje obavezno upisom slova "C" ili "*" u prvu kolonu. Tekst moze poceti od druge kolone. On ne utice na izvrsenje programa, i moze stajati bilo gdje u programu (ali ispred END naredbe). Kod prevodjenja programa prevodilac ne smatra naredbama i ne prevodi ove linije koda.

Naredbe se upisuju i izvrsavaju odozgo prema dole.

3.2. Dijagram toka programa

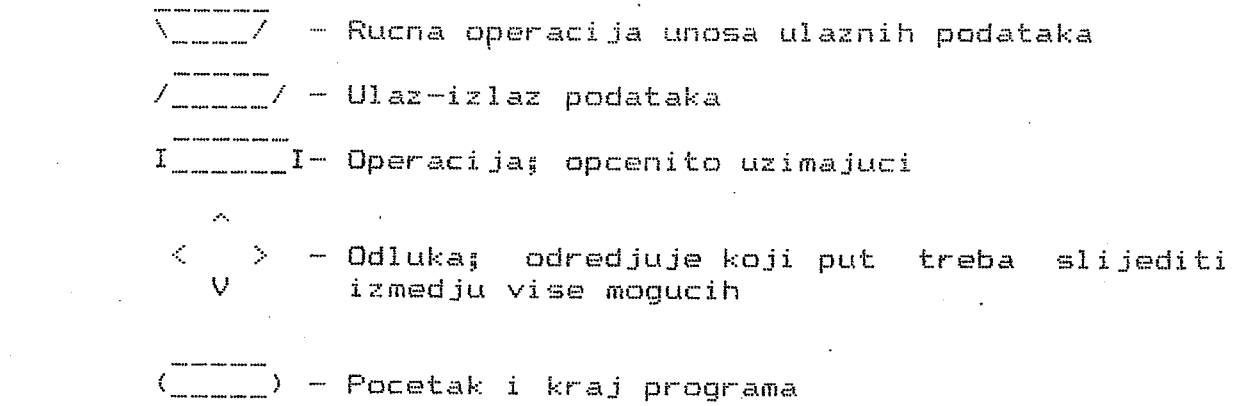
Prije nego se pristupi pisanju programa treba:
-Tocno definirati zadatak
-Izraditi dijagram toka programa (koji se naziva i: flow-chart, plan programskog toka, blok-dijagram, organigram).

Dijagram toka programa je graficki nacin prikaza funkcioniranja programa tj. graficki zapis ideja o logickom putu na koji ce se rijesiti odredjeni problem. On predocuje redosljed instrukcija koje ce kasnije pisati u programu. Pogodan je kod rijesavanja velikih i kompleksnih problema. Preporucuje se njegovo koristenje za programere pocetnike.

Postoje i druga sredstva za definiranje logike programa:
- program mozemo opisati rijecima, usmeno ili pismeno na prirodnom jeziku
- stabla odlucivanja
- pseudo kod
- Nassi-Shnauderman dijagrami (Chapin charts)
- tabele odlucivanja

koja su detaljno opisana i literaturi 18. Tabele odlucivanja se odlikuju najpovoljnijim karakteristikama ali ce logika programa biti predstavljena najrasprostranjenijim klasicnim sredstvom, dijagramima toka programa.

Princip dijagrama toka programa je da se naredbe smjestaju u simbole (likove, figure) raznih formata spojene linijama koje pokazuju smjer i redoslijed izvodjenja. Ovakvo odredjen redoslijed izvodjenja olaksava pisanje FORTRAN-naredbi. Simboli za crtanje dijagrama odvijanja operacija programa su propisani standardom (JUS A.FO.004 XII-1971.;Sluzbeni list SFRJ br. 58/1971.). Najcesce koristeni simboli jesu:



0 - Prikljucna tocka; mjesto izlaza ili ulaza u jedan drugi dio programa.

------ - Linija odvijanja operacija programa; povezuje ostale simbole, obavezno je crtati strelicu na kraju linije ukoliko je smjer toka odozdo prema gore ili zdesna na lijevo.

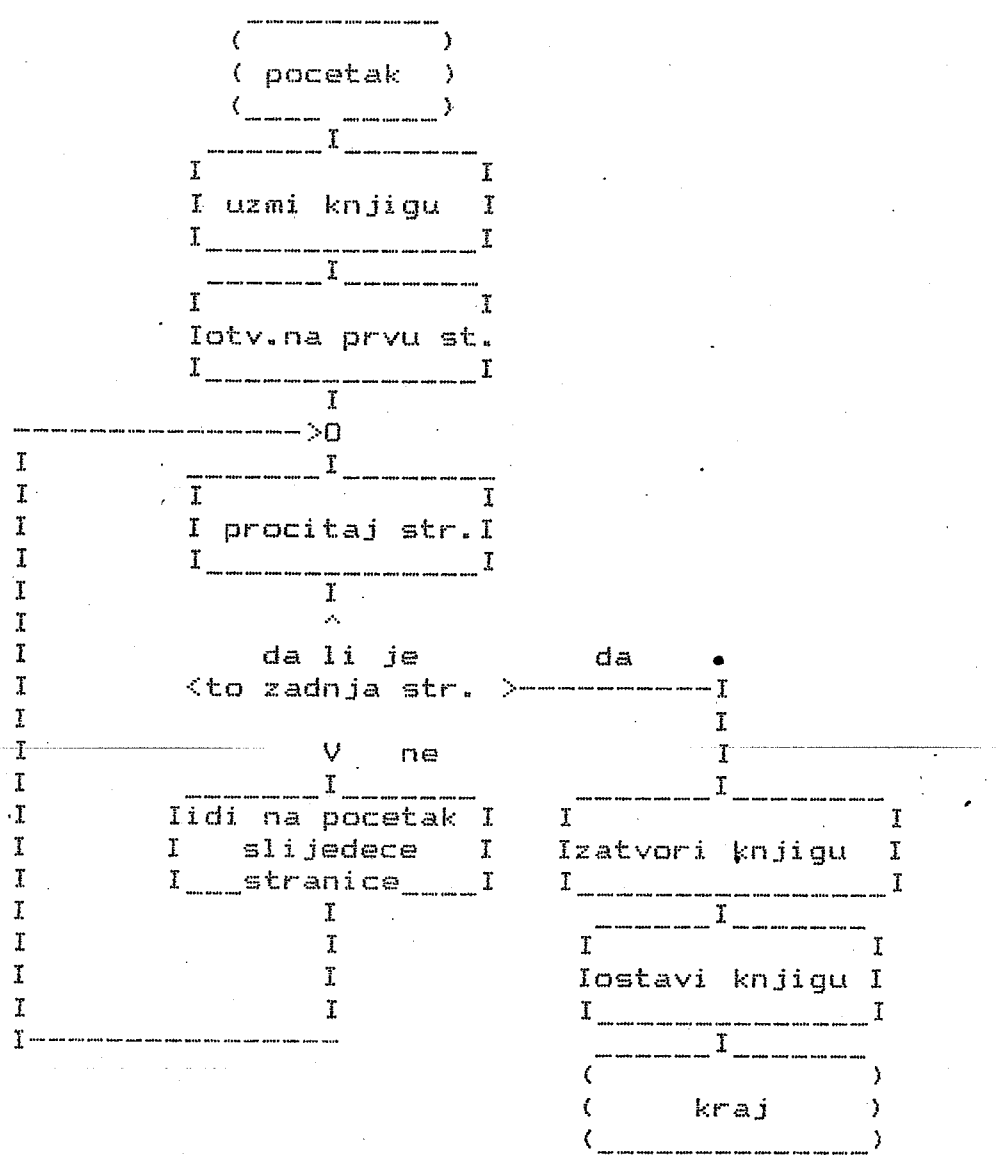
- Potprogram; mjesto u programu odakle se pozivamo potprogram na izvrsenje.

Analizom jedog jednostavnog primjera, kao sto je "citanje knjige", date su u obliku niza instrukcija potrebne operacije, a to su ujedno i upute koje definiraju realizaciju zadataka.

Zadatak: Procitaj knjigu.
Potrebne operacije i redoslijed izvrsenja operacija za realizaciju zadatka je slijedeci:

1. Uzmi knjigu
2. Otvori je na prvu stranicu
3. Procitaj stranicu
4. Ako je zadnja stranica, idi na instrukciju br.7
5. Idi na pocetak slijedece stranice
6. Idi na instrukciju 3
7. Zatvori knjigu
8. Ostavi knjigu

Izvrsenjem prve instrukcije prelazi se na slijedecu, ukoliko njeno izvrsenje ne zahtjeva skok na labelu druge instrukcije. Na osnovu tako ispisanih instrukcija nacrtan je dijagram toka programa.



Na slican nacin ce se rjesavati daljnji zadaci. Ovako rjesen zadatak je razumljiv programerima u bilo kom programskom jeziku jer se ovako dan dijagram toka programa da jednbostavno prevesti PASCAL, COBOL, PL/1 i dr. S obzirom na to mozemo zakljuciti da vjestina programiranja, odnosno rjesavanja problema pomocu programskih jezika, nije u pisanju linija koda programa vec u pronalazenju logickog puta kojim trebaju proci ulazni podacida dobijemo zeljeni rezultat. Odavdi i vazno upozorenje programerima, napose pocetnicima da se ne upustaju u pisanje linija koda programa dok logika rada programa nije jasno specificirana dijagramom toka programa.

3.3. Izvrsenje FORTRAN-skog programa na terminalu

Kako bi polaznici tecaja mogli svoje programe izvoditi na racunaru paralelno uz ucenje programa dana su kratka uputstva za ovaj rad s FORTRAN kompajlerom na racunaru UNIVAC 1100/60.

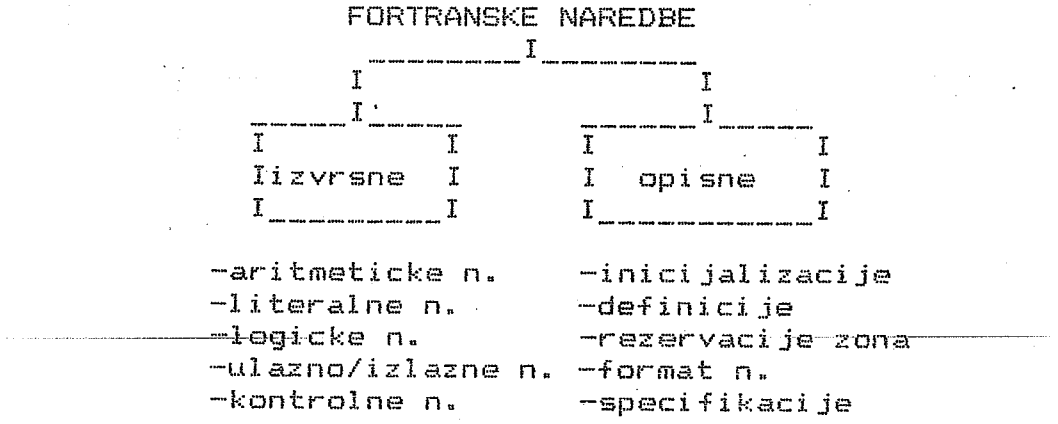
```
>broj terminala
>user-id/password
>@FTN,NCI   ime
   program
>@EOF
>     ENTERING USER PROGRAM
 rezultat rada programa
>END PROGRAM EXECUTION
```

npr.:

```
>S41M09
>AOMOST/TAJNA
>@FTN,NCI     A
>          PRINT *,' testiranje fortran programa '
>          PRINT *,' sin(pi/2)=',sin(3.14/2)
>@EOF
>     ENTERING USER PROGRAM
>testiranje fortran programa
>sin(pi/2)= .99999968
>END PROGRAM EXECUTION
```

Tehnologija izvodjenja ovisi o racunaru. Preporucuje se citaocu koji ima mogucnost rada s racunarom da se upozna s procedurama potrebnim za izvodjenje FORTRAN-programa i prakticno izvrsava zadatke koji slijede.

U daljnjem dijelu prirucnika potrebna je aktivnija uloga programera u savladavanju sintakse jezika i logike programiranja. Osnovna jedinica od koje je sacinjen svaki program je naredba. Sa aspekta funkcije u programu mozemo naredbe podijeliti na izvrsne i opisne.



-Izvrsne naredbe odredjuju koja se operacija treba izvrsiti i nad kojim podacima, pa prema tome predstavljaju akciju koju racunar treba sprovesti. U tu grupu spadaju sve naredbe za dodjeljivanje vrijednosti, kontrolne i ulazno/izlazne naredbe.

-Opisne naredbe definiraju: tip, raspored, inicijalne vrijednosti podataka, funkcije, vrstu potprograma i pruzaju sve dodatne informacije potrebne za izvrsenje naredbe koje se odnose na to kako treba izvrsiti odredjenu akciju, ili daju informaciju o programu u cjelini.

Za tvorbu FORTRAN naredbi koriste se specijalne oznake, poznate kao sluzbene rijeci (vidi Dodatak B.).

Mada svaka naredba moze imati labelu, samo labele izvrsnih naredbi (izuzevsi naredbe ELSE IF i ELSE) i FORMAT naredba mogu biti upotrebljene u programu. Neizvrsne (opisne) naredbe su: DIMENSION, COMMON, EQUIVALENCE, PARAMETAR, EXTERNAL, INTRINSIC, IMPLICIT, INTEGER, REAL, DOUBLE PRECISION, COMPLEX, CHARACTER, LOGICAL, SAVE, DATA, FORMAT, naredbe za definiranje funkcija, PROGRAM, FUNCTION, SUBROUTINE, ENTRY, BLOCK DATA. Naredbe EXTERNAL i INTRINSIC nisu obuhvacene ovim tekstom.

4. NAREDBE ZA DODIJELJIVANJE VRIJEDNOSTI

Osnovna im je namjena da imenima varijabli pridruze vrijednosti izraza i sacuvaju ih za daljnje koristenje u programu.

Tipovi naredbi za dodijeljivanje vrijednosti su:

4.1. Aritmeticke naredbe
4.2. Literalne naredbe
4.3. Logicke naredbe

4.1. Aritmeticke naredbe

Aritmetickom naredbom se simbolu jedne varijable ili clanu matrice, koji su tipa INTEGER, REAL, DOUBLE PRECISION ili COMPLEX, dodjeljuje rezultat aritmetickog izraza.

opci oblik:

V=a

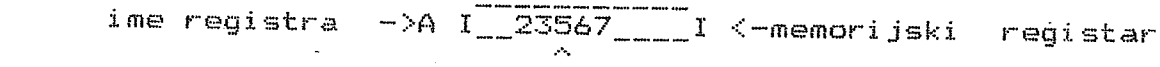
V je ime varijable ili clana matrice
a je aritmeticki izraz

Kod izvsenja programa vrijednost "a" prelazi u brojevnu vrijednost i dodjeljuje se varijabli "V". Na lijevoj strani znaka "=" moze biti samo jedna varijabla. Vrijednost varijable se cuva u memorijskom registru (mjestu sposobnom da upamti podatak) i moze biti dalje koristena. Ako je tip rezultata izraza "a" razlicit od tipa varijable "V", tada se mijenja tip rezultata u tip varijable. Ako bi rezultat izraza bio tipa CHARACTER ili LOGICAL, pojavljuje se greska.

npr.: Ako je dana aritmeticka naredba

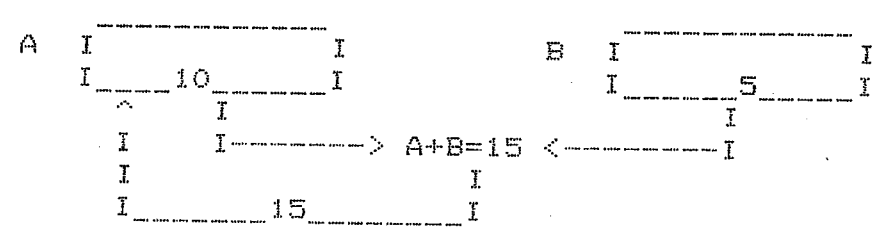
A=23567

to znaci da jednom memorijskom registru koji se imenuje A je dodijeljena vrijednost 23567. U memorijski registar je upisana vrijednost 23567 dok se nekom aritmetickom naredbim ne izmjeni vrijednost varijable A ili ne zavrsi rad programa.



I_ sadrzaj registra

Naredba A=A+B nalaze racunaru da uzme sadrzaj memorijskih registara A i B i rezultat smjesti u registar A. Znak "=" nema isti smisao kao u matematici.



Primjeri aritmetickih naredbi:

a) I=18
b) A=B/C nije dobro A+C=B
c) A=6./2. ==>A=3.
d) I=A <-integer velicina jednaka je cijelom djelu realne velicine (bez decimala)
e) A=I <-realna velicina jednaka je integer velicini
f) NOV=2I+3C

```
g)   AB=C**2+AB+4.36
h)   I=  J+A**2-4
i)   BRZINA =PUT/VRIJEM
j)   A=I/3-4*B    (za I=10 i B=3.5)=> A=-11.0
       ---
        ^_____  integer=3
```

Ako se u nekom aritmetickom izrazu koristi varijabla cija vrijednost nije inicijalizirana ranije u programu, onda se implicitno podrazumijeva da je vrijednost varijable jednaka 0.

STANDARDNE MATEMATICKE FUNKCIJE

Niz matematickih funkcija(npr.$\sqrt{2}$, trigenometrijske, logaritamske, hiperbolne, apsolutne vrijednosti, ekstremne vrijednosti, gama-funkcije itd.) postoje u FORTRAN-u kao gotovi programi i kao takve ih mozemo koristiti u programu. Ove i ostale funkcije respolozive programeru popisane su u tabeli Standardnih funkcija (vidi Dodatak C.). Neke od tih funkcija su:

za funkciju		pisemo u programu	
$y=\ln x \quad (x > 0)$	--->	y=alog(X)	
$y=\log_{10} x \quad (x > 0)$	--->	y=alog10(x)	
$y=e^{x}$	--->	y=exp(X)	
$y=\sqrt{x} \quad (x >= 0)$	--->	y=sqrt(x)	(square root)
$y=\sqrt[3]{x}$	--->	y=cbrt(x)	(cube root)
$y=\sqrt[7]{x}$	--->	y=x **(1/7)	
y=tanx	--->	y=tan(X)	x u radijanima
y=|x|	--->	y=iabs(x)	ili y=abs(x)
	itd.		

Poziv funkcije moze se u aritmetickom izrazu pisati na svakom mjestu gdje je potrebno izracunati vrijednost funkcije. Iza imena funkcije u zagradama upisujemo argument (aritmeticki izraz) cija se vrijednost racuna prije racunanja vrijednosti funkcije.

Ostale matematicko-statisticke funkcije nalaze se u paketu programa "MATHSTAT PACK" na racunaru.

npr.:

1.Napisi u obliku fortranske aritmeticke naredbe

a. $n= 2\cos y + x\sin(z+3.14)$

b) $u=\frac{1}{2}\cos(y)*\sin(z)$

c) $y=\frac{1-\arccos x}{1+\arcsin x}$ rjesenje: y=(1.-acos(x))/(1.+asin(x))

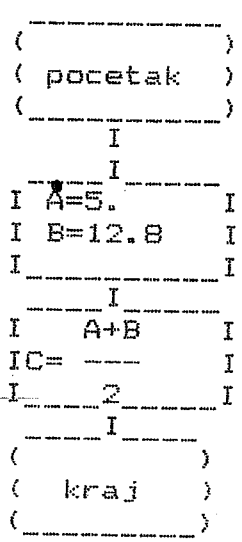
d)y=ln|tgx+ctgx| rjesenje: y=alog(abs(tan(x)+cotan(X)))

e) $y=e^{2arctg(x+2y)}$ rjesenje: y=exp(2.*atan(X+2*Y))

Zadatak: Nacrtati blok dijagram i napisati program za izracunavanje srednje vrijednosti broja A=5,0 i B=12,8 te je smjestiti u varijablu C.

Rjesenje:

Dijagram toka programa (DTP)



Program:

```
REAL A,B,C
A=5.
B=12.8
C=(A+B)/2
```

4.2. Literalne naredbe

Literal naredbom se dodijeljuje literala varijabli ili literala clanu materice vrijednost literal izraza.

opci oblik:

V=k

gdje je "V" literal varijabla i "k" literal izraz.

npr.

IMENA='MARKO'//VAR//' DARKO'//VAR1//'ZLATKO'

Varijable VAR iVAR1 su tipa character.

Literal podtekst je dio literal varijable ili clana matrice. Opci zapis podteksta je V(e1:e2) gdje je "V" ime literal varijable, "e1" i "e2" jesu cijelobrojni izrazi, koji odredjuju lijevu i desnu poziciju karaktera u varijabli. Opci oblik podteksta za element matrice je A(a,b,...)(e1:e2) gdje je A(a,b,...) element matrice "A".
Vrijednosti "e1" i "e2" moraju zadovoljiti uvijet 1<=e1<=e2<=len, gdje je "len" duzina literal varijable ili clana matrice u pozicijama. Parametre "e1" i "e2" nije obavezno pisati.

npr.

A(2:4) oznacava literal podtekst od druge do cetvrte pozicije varijable "A".

B(4,3)(1:6) oznacava literal podtekst od prve do seste pozicije clana cetvrtog reda i treceg stupca matrice "B".

Ako je e1 ispusten, uzima se da je e1=1, ako je e2 ispusten uzima se da je e2=len. Ako oba e1 i e2 ispusteni tad je V(:)=V i A(a,b,...)(:)=A(a,b,...).

npr.
1. Ako je V='DAVID' i C=V(3:5) tad je C='VID'
2. CHARACTER*4 C1,C2(2,2)
   C2(1,1)(I:I+1)=C1(3:4)

4.3. Logicke naredbe

Logickom naredbom dodijeljujemo logickoj varijabli ili clanu matrice vrijednost logickog izraza.

opci oblik:

V=l

gdje je "V" logicka varijabla i "l" logicki izraz.

npr.

LOG1=(L.OR.L1).AND.(B.OR.B1)

5. K O N T R O L N E N A R E D B E

Normalno izvrsenje programske jedinice zapocinje sa prvom izvrsnom naredbom u programu, i nakon njenog izvrsenja izvrsava se prva slijedeca naredba. Ovaj postupak se nastavlja sve dok i poslijednja izvrsna naredba u programu ne bude izvrsena. Kontrolne naredbe mijenjaju ovakav normalan redoslijed izvrsenja. Neke od njih bezuvijetno mijenjaju ovaj poredak, a neke to izvrsavaju u ovisnosti o rezultatu testa sadrzanog u naredbi. Kontrolne naredbe su: END, STOP, PAUSE, CONTINUE, GO TO, IF, blok IF, DO, CALL, RETURN. Naredbe CALL i RETURN su opisane u poglavlju POTPROGRAMI.

5.1. END naredba

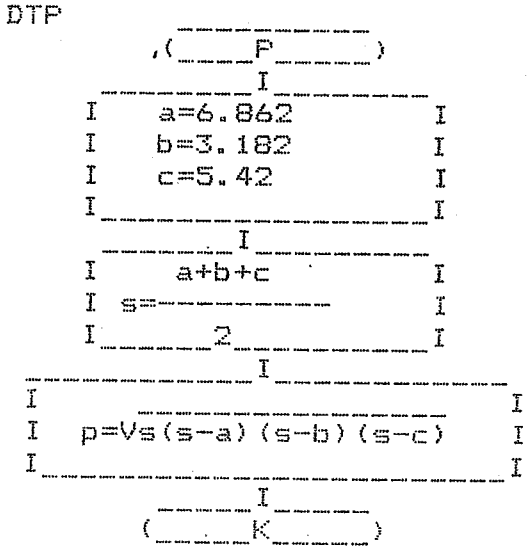
Za oznaku fizickog kraja programa ili potprograma obavezno je pisati END kao posljednju naredbu u programu. To je informacija prevodiocu da prethodne naredbe shvati kao jednu grupu (programsku jedinicu). Time je rad programa ili potprograma zavrsen. END je sluzbena rijec.

ZADATAK: Izracunati povrsinu trokuta ako je zadana duzina stranica : a=6.862, b=3.18 i c=5.42. Za rjesenje zadatka koristiti Heronovu formulu:

$$P=\sqrt{s(s-a)(s-b)(s-c)}$$

$$s=\frac{a+b+c}{2}$$

Rjesenje:




```
Program

REAL A,B,C,S,P
A=6.962
B=3.18
C=5.42
S=(A+B+C)/2
P=SQRT(S*(S-A)*(S-B)*(S-C)
END
```

KOMENTAR

Radi povecanja jasnoce programa koristi se znak "C" ili "*" (comment) u prvoj koloni za upis slobodnog teksta ili prazne linije u programu.

npr.:

```
 1   5 6 7                                                  7     8
I_____I_I_____________________________________________I2I____OI
IC    I I                                             I I     I
IC    I I                                                     I
I*      I          RACUNANJE POVRSINE TROKUTA                 I
IC      I     pomocu HERONOVE formule                         I
I       I REAL A,B,C,S,P                                      I
I       I A=6.862                                             I
I       I B=3.18                                              I
I       I C=5.42                                              I
IC      I                                                     I
I       I              S=(A+B+C)/2                            I
IC  HERONOVA FORMULA                                          I
IC      I                                                     I
I       I P=SQRT(S*(S-A)*(S-B)*(S-C))                         I
IC      I                                                     I
I       I      END                                            I
```

ZADATAK: Kosinusovim pouckom izracunaj stranicu trokuta, ako su stranice b=7.3, c=8.3 i kut medju njima alfa=2 radijana.
Formula za kosinusov poucak:

$$a^2 = b^2 + c^2 - 2bc\cos(alfa)$$

PRINT naredba

Spada u grupu ulazno/izlaznih naredbi a omogucuje korisniku ispis rezultata obrade ili ispisa proizvoljnog teksta. Primjetimo da u prethodnim primjerima ne bi vidjeli rezultat obrade podataka.

opci oblik:

```
PRINT *,varijable,'TEKST',.....
```

npr.:

Ako je A=6.862, B='T', C=3, S=2, P=12 ispisite vrijednosti varijabli primjenom PRINT naredbe.

naredba	rezultat
PRINT *,A	6.862
PRINT *,'REZULTAT:'	REZULTAT:
PRINT *,'A=',A	A=6.862
PRINT *,A,B,C,S,P	6.862 T 3 2 12
PRINT *,'A=',A,' B=',B,' C=',C	A=6.862 B=T C=3
PRINT *,'S=',S,';P=',P	S=2;P=12

U zadnja dva primjera, radi preglednosti, ispred vrijednosti varijable ispisano je ime varijable, sto se i preporucuje.

Zadatak: 1. Ukljuciti naredbu PRINT u prethodnim zadacima da se dobije ispis rezultata aritmetickih naredbi.
2. Kako bi izgledao rezultat rada programa?

```
PRINT*,'I=',I,'A=',A
END
```

ako je I=153 i A='TEXT'

5.2. STOP naredba

Nailaskom na naredbu STOP vrsi se prekidanje izvodjenja programa.

opci oblik:

STOP n ili STOP 'poruka'

gdje je:

n - znak ili cijelobrojna konstanta duljine od 1 - 5 znakova

poruka - litelarna konstanta pod apostrofima do maksimalno 124 karaktera

Nakon izvrsenja naredbe STOP program se ne moze nastaviti, a ispisuje se poruka STOP n ili STOP poruka.

npr.:

Za program	Rezultat
A = 5 B = 12.8 PRINT *,A,B STOP 10 C=(A+B)/2 PRINT *,C END	5.0000000 12.8000000 STOP 10 (Program ne bi racunao aritmeticku sredinu.)

npr.: STOP "GRESKA"

Sluzi za zaustavljanje rada programa ako se pojavila greska u grani programa obiljezenoj rijecju GRESKA.

5.3. PAUSE naredba

Sluzi za privremeno zaustavljanje izvrsenja programa u terminalskom (DEMAND) radu i oznaku mjesta u programu gdje je to zaustavljanje izvrseno. Program se moze nastaviti pritiskom na odgovarajucu dirku (XMIT, RETURN, ENTER itd.).

opci oblik:

PAUSE n
ili
PAUSE 'poruka'

gdje je:

n - znak ili cijelobrojna konstanta duljine od 1 - 5 znakova

poruka - litelarna konstanta pod apostrofima do maksimalno 124 karaktera

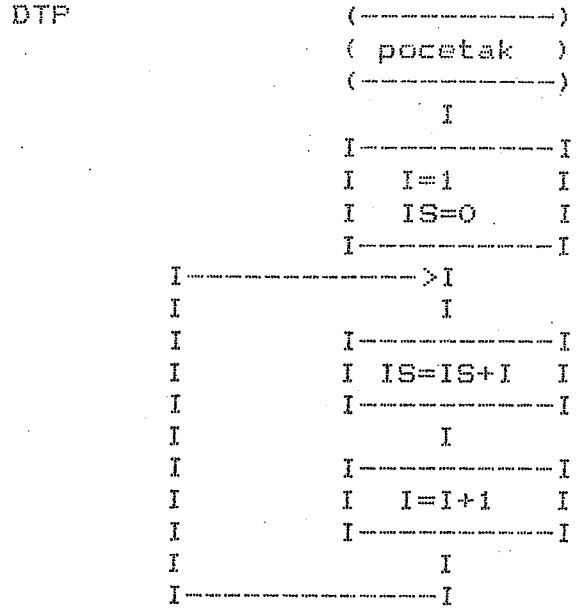
Ako su u programu dane naredbe PAUSE onda ce program ispisati:

naredba u programu	rezultat (dejstvo naredbe)
PAUSE 2	PAUSE 2
PAUSE	PAUSE 00000
PAUSE 'GRANA 2'	PAUSE GRANA 2
PAUSE 123456	PAUSE 123456

U Batch modu se nastavlja izvrsenje programa bez obzira na naredbu PAUSE.

U Demand modu se izvrsenje nastavlja pritiskom na drirku XMIT, RETURN, ENTER i dr. Naredba PAUSE omogucuje zaustavljanje rada programa dok se ne pregledaju medjurezultati i u ovisnusti o njihovoj tocnosti prekine ili nastavi daljnje izvrsenje programa.

npr.2: Napisati program koji ce zbrajati prirodne brojeve. GO TO naredba omogucuje ponavljanje grupe naredbi.



Program verzija 1

```
      I=1
      IS=0
5     IS=IS+I
      PRINT *,'I=',I
      PRINT *,'S=',S
      I=I+1
      GO TO 5
      END
```

Program verzija 2

```
      INTEGER S,I
      I=1
5     CONTINUE
      S=S+I
      PRINT *,'I=',I,'S=',S
      I=I+1
      GO TO 5
      END
```

5.5.2. Izracunati GO TO

Omogucava granjanje programa u vise pravaca

opci oblik:

GO TO (n1,n2, ... nm),k

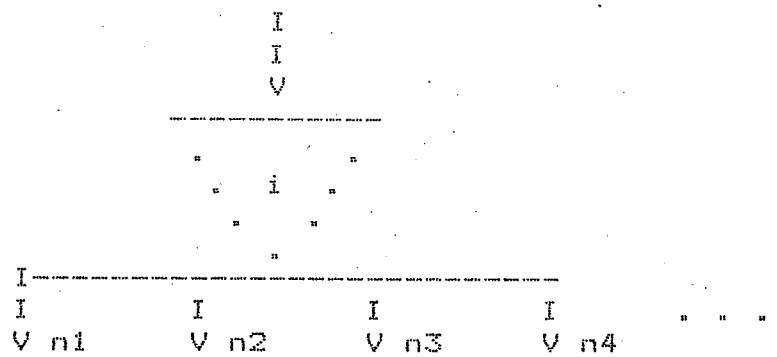
gdje je:

k je cijelobrojana varijabla koja je dobila vrijednost prije nailaska na izracunatu GO TO naredbu u granicama 0 < k < m

n1,n2, ... nm su brojevi labela na koje se prenosi kontrola po pravilu:

ako je k=1 kontrola se prenosi na naredbu s labelom n1
ako je k=2 kontrola se prenosi na naredbu s labelom n2
ako je k=m kontrola se prenosi na naredbu s labelom nm
ako je k=0 kontrola se prenosi na slijedecu naredbu

Graficki se ova naredba da predstaviti kao:



5.4. CONTINUE naredba

Sluzi za adresiranje mjesta u programu.

opci obli:

```
n      CONTINUE
```

gdje je:

n	je labela naredbe CONTINUE. . To je broj od 1 do 99999
CONTINUE	je sluzbena rijec

Naredba moze biti smjestena bilo gdje u programu a omogucava prijelaz na slijedecu izvrsnu naredbu.

5.5. Naredbe za bezuvjetni prelazak(GO TO)

Sluze za izmjenu redoslijeda izvrsenja naredbi, te se ne izvrsava slijedeca naredba u sekvenci (izuzev ako se na nju ne prenosi kontrola).

Postoje tri vrste "GO TO" naredbi:

```
Bezuvjetni  GO TO
Izracunati  GO TO
Asajnirani  GO TO
```

5.5.1. Bezuvjetni GO TO

Omogucava izmjenu redoslijeda izvsenja naredbi:

opci oblik:

```
GO TO n
```

gdje je:

n	je labela (vidi naredbu CONTINUE)

Prva izvrsna naredba iza GO TO naredbe mora imati labelu. Pri nailasku na GO TO naredbu kontrola se prenosi na naredbu s labelom n, bez obzira gdje se ona u programu nalazi(ispred ili iza GO TO naredbe).

Na istu naredbu se moze doci sa vise mjesta iz programa.

npr.1: Dolazak na naredbu s labelom 15 iz raznih mjesta u programu.

```
      GO TO 15
        . . .
        . . .
        . . .
15    CONTINUE
        . . .
        . . .
        . . .
      GO TO 15
```

npr.1: Ako je k= 5 i ako je dana naredba

GO TO(10,5,15,204,14,7),k

kontrola odvajanja programa prenijeti ce se na naredbu s labelom 14 jer je broj 14 peti po redu u zagradama.

5.6. Naredbe za uvjetni prelazak (IF)

Omogucuju uvjetni prijenos kontrole na neku izvsnu naredbu u ovisnosti o vrijednosti uvjeta.

Postoje tri vrste ovih naredbi:

- Aritmeticka naredba grananja
- Logicka naredba grananja
- Blok naredba grananja

5.6.1. Aritmeticka naredba grananja

Omogucava prijenos kontrole programa na neku naredbu u ovisnosti o vrijednosti aritmetickog izraza.

Opci oblik:

IF(aritmeticki izraz) n1,n2,n3

gdje su:

n1,n2,n3 ——— su labele na koje se prenosi kontrola po pravilu:

ako je vrijednost arit.izr. < 0 idi na naredbu s labelom n1
ako je vrijednost arit.izr. = 0 idi na naredbu s labelom n2
ako je vrijednost arit.izr. > 0 idi na naredbu s labelom n3

Ako vrijednost neke od labela nije upisana a kontrola je usmjerena na neupisanu labelu, izvrsava se slijedecu naredbu.

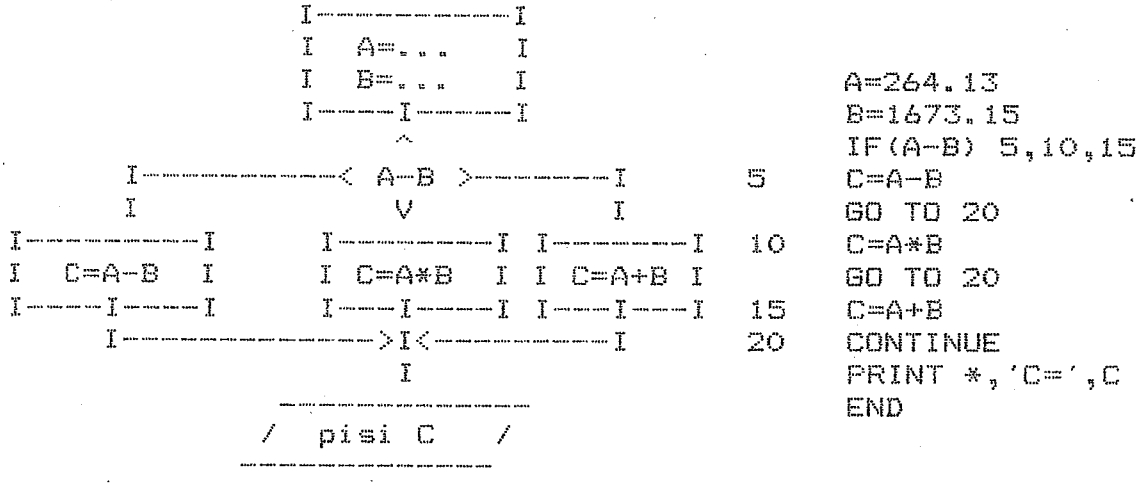
npr.1: Dano je nekoliko primjera mogucih izgleda naredbe.

```
IF(5-J) 1,2,3
IF(A+3*B-2.) ,15,20
IF(A(7,1)) 100,200,
IF(ARR(I,J))99,,
```

npr.2: Napisati opci program za racunanje vrijednosti funkcije C ako je za bilo koje vrijednosti A i B definirana funkcija:

```
     { A+B        za A > B
C=   { A*B        za A = B
     { A-B        za A < B
```

Neka je A=264.13 i B=1673.15 tada program imà izgled:




```
A=264.13
B=1673.15
IF(A-B) 5,10,15
C=A-B
GO TO 20
C=A*B
GO TO 20
C=A+B
CONTINUE
PRINT *,'C=',C
END
```

5.6.2. Logicka naredba grananja

Omogucava grananje programa u ovisnosti o vrijednosti logickog izraza.

opci oblik:

IF(l)S

gdje je:

S je izvrsava naredbu razlicito od IF ili DO naredbe

l je logicki izraz

Ova naredba prenosi kontrolu po pravilu:

ako je vrijednost l=.TRUE. izvrsava se naredba S. Ako je S naredba grananja, kontrola se prenosi na drugi dio programa.

ako je vrijednost l=.FALSE. ne izvrsava sa naredba S vec prva slijedeca naredba iza IF naredbe.

npr.1: Ako su dane naredbe:

```
IF(K.LT.3.14)  K=K+3.14
Y=SIN(K)
```

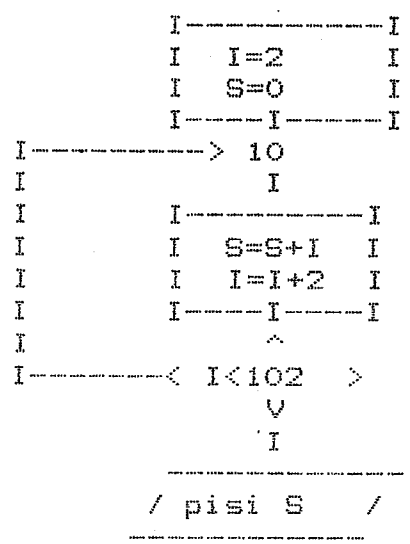
za K<3.14 izvrsava se prvo naredba K=K+3.14 pa onda naredba Y=SIN(K) a za K>=3.14 izvrsava se samo naredba Y=SIN(K).

npr.2: Primjer logickih naredbi grananja:

```
a) IF(A.LE.B+C)  A=ALOG(B+C)
b) IF(TEKST.EQ.'YES') GO TO 5
c) IF(.NOT.(TEKST.NE.'YES')) GO TO 5
```

Primjetimo da je rezultat isti u b) i c) primjeru.

npr.3: Naci sumu parnih brojeva od 2 do 100. Ispitivanje kraja sumiranja vrsiti logickim naredbama grananja.




```
      INTEGER S
      I=2
      S=0
10    S=S+I
      I=I+2
      IF(I.LT.102) GO TO 10
      PRINT *,'S=',S
      END
```

5.6.3. Blok naredba grananjna

Omogucava grananje programa u jedan od dva bloka u ovisnosti o vrijednosti logickog izraza.

Opci oblik:

```
IF(l) THEN
  . . .
  . . .
  . . .
ELSE
  . . .
  . . .
  . . .
END   IF
```

gdje je:

l je logicki izraz

Blok IF naredba omogucuje strukturno programiranje operacija tipa selekcije. Ako je l istinit, tada se izvrsava blok naredbi do sluzbene rijeci ELSE a nakon toga kontrola se prenosi na naredbu iza sluzbene rijeci END IF. Ako je l lazan, izvrsava se blok naredbi izmedju sluzbenih rijeci ELSE i END IF a nakon toga kontrola se prnosi na naredbu iza sluzbene rijeci END IF. Radi preglednosti programa preporucava se pisanje naredbi u bloku nekoliko kolona udesno u odnosu na sluzbene rjeci IF, ELSE i END IF . Unutar IF bloka dozvoljena je IF blok naredba. Preporucuje se upotreba ove naredbe.

npr.1: Primjer pisanja IF blok naredbe.

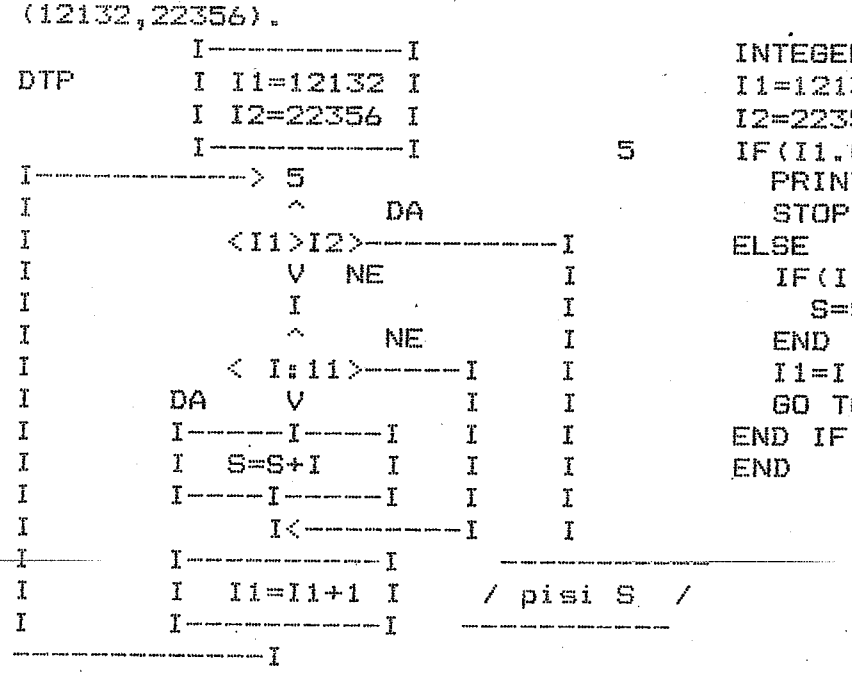
```
IF(K.GE.132) THEN
   K=0
ELSE
   K=K+1
END IF
PRINT *,K
END
```

Ako je k >= od 132 program ce promjeniti vrijednost K u K=0 i ispisati vrijednost. Ako je K < 132 vrijednosti K se dodaje 1 i ispisuje nova vrijednost.

Z A D A C I:

U svim zadacima preporucuje se najprije nacrtati dijagram toka programa i na osnovu njega napisati program.

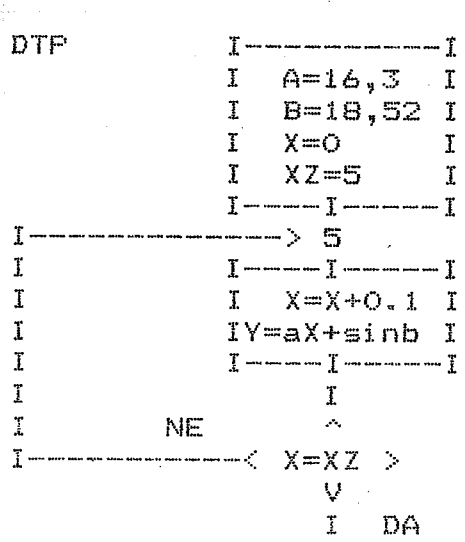
1. Naci koliko je brojeva djeljivih sa 11 u intervalu (12132,22356).




```
      INTEGER S
      I1=12132
      I2=22356
    5 IF(I1.GT.I2)THEN
        PRINT *,' IMA IH',S
        STOP ' GOTOVO'
      ELSE
        IF(I1.EQ.I1/11*11)THEN
          S=S+1
        END IF
        I1=I1+1
        GO TO 5
      END IF
      END
```

2. Izracunati vrijednost funkcije Y=ax+sinb za a=16,3 i b=18,52. Za vrijednost 0<x<=5 s korakom dx=0,1.

Program napisan pomocu logicke IF naredbe.



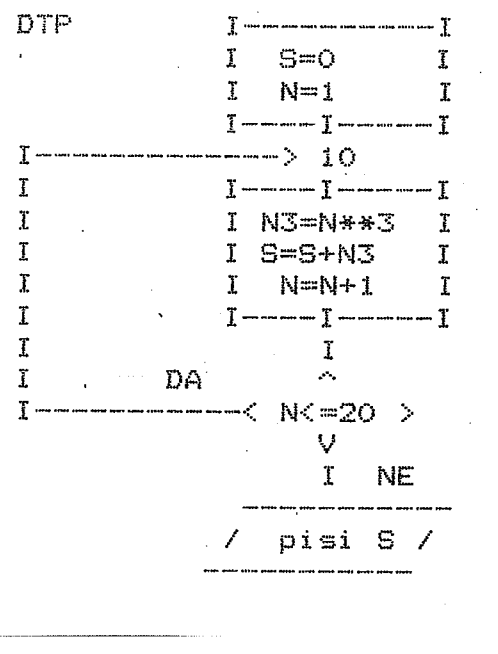

```
      A=16.3
      B=18.52
      XZ=5
   10 X=X+0.1
      Y=A*X+SIN(B)
      PRINT *,'X=',X,'Y=',Y
      IF(X.NE.XZ) GO TO 10
      END
```

Program napisan pomocu aritmeticke IF naredbe

```
      A=16.3
      B=18.52
   10 X=Y+0.1
      IF(X-55),,15
      Y=A*X-SIN(B)
      PRINT *,'X=',X,'Y=',Y
      GO TO 10
   15 CONTINUE
      END
```

3. Napisati program koji racuna sumu od 20 clanova reda koristeci logicki IF.

$$1^3 + 2^3 + 3^3 + \ldots + 20^3 = SUMA(N^3)_{N=1}^{20}$$




```
      INTEGER SUM
      SUM=0
      N=1
      N=N+1
      IF(N.LE.20)GO TO 10
      PRINT *,'SUMA=',SUM
      END
```

4. Napisati program koji racuna sumu od 20 clanova reda koristeci aritmeticki IF.

$$1 + 1/2 + 1/4 + 1/8 + \ldots = SUMA\left((1/2)^n\right)_{n=0}^{20}$$

```
      SUM=0
      N=1
      CL=1.0
10    SUM= SUM+CL
      CL=CL/2
      N= N+1
      IF(N-20)10,10,
      END
```

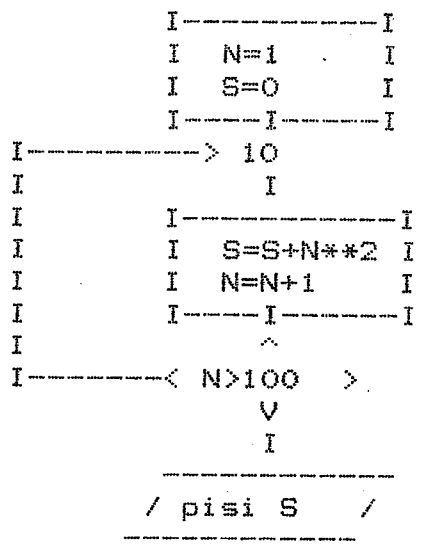
5. Trazi se y(x) za 1<=x<=2,5 u koracima od dx=0,1 ako je:

$$Y = \begin{cases} 0,7x + 0,62 & \text{za } x <= 1,7 \\ 0,9x + 0,41 & \text{za } x >= 1,7 \end{cases}$$

6. Ispitati i ispisati koji su cijeli brojevi u intervalu [10,30] dijeljivi sa brojem 7.

7. Izracunati kvadrate prirodnih brojeva od 1 do 100 i naci sumu svih kvadrata.

DTP




```
      PROGRAM VERZIJA 1
           N=1
           M=0
    5      M=M + N**2
           N=N+1
           IF(N-100)5,5,6
    6      CONTINUE
           PRINT *,'SUMA=',M
           END
```

```
      PROGRAM VERZIJA 2
        N=1
        M=0
   10   IF(N.LE.100)THEN
           M=M+N**2
           N=N+1
           GO TO 10
        ELSE
           PRINT *,'SUMA=',M
        END IF
        END
```

8. Rijesiti zadatak 7. tako da se suma racuna od 100 do 1.

9. Napisati tablicu mnozenja do 10, da izlazini rezultat bude ispisan u obliku:

```
1*1=1
1*2=2
1*3=3
 . . .
 . . .
1*10=10
2*1=2
 . . .
```

```
           N=1
   15      K=1
    5      IP=N*K
           PRINT *,N,'*',K,'=',IP
           K=K+1
           IF(K-10)10,,20
   10      GO TO 5
   20      N=N+1
           IF(N-10) 15,15,25
   25      CONTINUE
           END
```

```
      Napisati sa IF THEN
           N=1
    5      K=1
   10      IF(K.LE.10) THEN
              IP=N*K
              PRINT *,N,'*',K,'=',IP
              K=K+1
              GO TO 10
           ELSE
              N=N+1
              GO TO 5
           END IF
           END
```

10. Izracunati faktorijele do 10!, a izlazni rezultat ispisati u obliku:

```
1!=1          Formula za racunanje faktorijela je:
2!=2                n!=(n-1)!*n
3!=6
. . .
```

11. Naci sve brojeve od 100 do 200 koji su dijeljivi sa 13. Za ispitivanje djeljivosti koristi aritmeticku IF naredbu :

```
IF(N-N/13*13)10,,20
```

12. Izracunati potencije broja pet (5) za eksponent iz domene [0,10]. Ispisati vrijednost eksponenta i funkcije.

$$F(x)=5^{x}$$

x je element iz [0,10] i x je cio broj.

13. Izracunati vrijednost funkcije

$$y=1-e^{x}\sin 2x+\log(\cos^{2} x)\,tgx$$

za vrijednost argumenta -5.34<=x<+5.34 . Argument mijenjati u intervalima po 0,1 s tim da se za tocku x=0 ne racuna vrijednost funkcije.

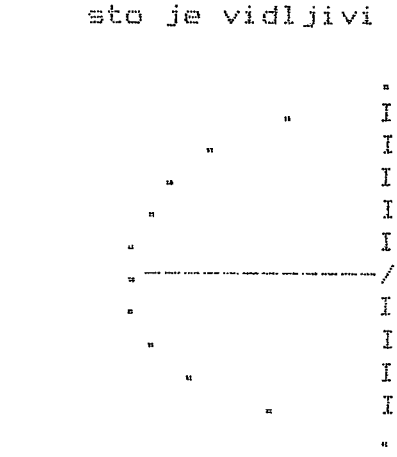
14. Napisati program koji pretvara sate, minute i sekunde u decimalno vrijeme. Zadano vrijeme je 04:36:07 .

```
      XSAT=04.
      XMIN=36.
      XSEC=07.
C     Izracunavanje decimalnog vremena
      DVR = XSAT + XMIN / 60. + XSEC / 3600.
      END
```

15. Napisati program koji pretvara kut u stupnjevima u radijane. Zadani kut je: ALFA = 20 32' 55". Za rjesavanje ovog zadatka potrebna je vrijednost konstante PI. Do nje dolazimo koristeci standardne fukcije. Pomocu funkcije npr. tg(x) za koju vrijedi formula:

$$tg\left(\frac{pi}{4}\right)=1$$

sto je vidljivi iz jedinicne kruznice




```
Program:

XSTU=20.
XMIN=32.
XSEC=55.
PI=4.0*ATAN(1.0)
XMIN=32.+XSEC/60.
XSTU=20.+XMIN/60.
RKUT=XSTU*PI/180.0
END
```

16. Izracunati potencije brojeva od 1 do 5 za eksponent od 0 do 10 (Bolje je pomocu DO petlje koja se obradjuje u nastavku teksta).

17. Odredite bar jedan pravokutni trokut s cijelobrojnim stranicama. Stranice trokuta varirati od 1 do 1000.

18. Od 20 realnih brojeva izbrojati koliko ih je vecih od 5,72. Proizvoljno odabrati bilo kojih 20 brojeva.

19. Za dane koordinate dvije tocke u ravnini izracunati njihovu udaljenost. Proizvoljno odabrati koordinate tocaka u ravnini. Udaljenost tocaka se racuna po formuli:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

5.7. DO naredba

DO naredba omogucuje strukturno programiranje operacija tipa iteracije.

opci oblik:

```
DO n i=m1,m2,m3
```

ili

```
DO n i=m1,m2
```

gdje je:

n je labela naredbe iza DO naredbe (a unutar programske jedinice) a koja je zadnja u sekvenci naredbi koje ce se visestruko izvoditi.

i je cijelobrojna, realna varijabla ili vbarijabla s dvostrukom preciznoscu.

m1,m2,m3 su konstante, varjable ili izrazi tipa INTEGER, REAL ili DOUBLE PRECISION (ne matrice), gdje je m1, pocetna vrijednost varijable "i", m2 krajna vrijednost varijable "i", m3 korak porasta varijable "i". Ako m3 nije naveden podrazumjeva se da je m3=1 (ne smije biti m3=0).

Nacin izvrsenja DO naredbe:

Sekvenca naredbi izmedju DO naredbi i naredbe s labelom n izvrsava se prvi put s vrijednoscu i = m1 ako je m1 <= m2. Ako je m1 > m2 sekvenca naredbi se ne izvrsava, a kontrola se prenosi na prvu naredbu iza naredbe s labelom n. Nakon izvrsavanja svih naredbi u sekvenci racuna se i=i+m3. Ako je nova vrijednost i>m2 prekida se izvrsavanje sekvence a u protivnom se izvrsava sekvenca naredbi, povecava vrijednost i za m3 i ispituje da li je i>m2. Izvrsenje naredbi u sekvenci se prekida kada vrijednost varijable i bude veca od m2. Naredbe u sekvenci se jos nazivaju DO petlja ili DO ciklus.

Naredbe na kojoj DO petlja ne smije zavrsiti jesu: GO TO, IF, ELSE, END IF, RETURN, STOP, END, DO, FORMAT.

U jednoj vanjskoj DO petlji moze biti vise unutrasnjih DO petlji. Unutrasnja DO petlja se potpuno izvrsava za jednu promjenu indeksa vanjske DO petlje. Dozvoljeno je imati maksimalno 25 DO petlji jednu u drugoj. DO petlje se ne smiju preklapati, kao npr:

DOZVOLJENO

```
------>      DO 5 I=1,10
I                ...
I                ...
I                ...
I  I---->    DO 10 J=1,20
I  I             ...
I  I             ...
I  I             ...
I  I-> 10    CONTINUE
I                ...
I                ...
I                ...
--->   5     CONTINUE
```

NIJE DOZVOLJENO

```
------->     DO 5 I=1,10
I                ...
I                ...
I                ...
I---------->  DO 10 J=1,20
I  I             ...
I  I             ...
I  I             ...
I  --> 5     CONTINUE
I                ...
I                ...
I                ...
I-----> 10   CONTINUE
```

Jedna naredba moze zatvoriti vise DO petlji. U DO petlji se ne treba mijenjati vrijednost i, m1, m2, m3.

nije dobro:

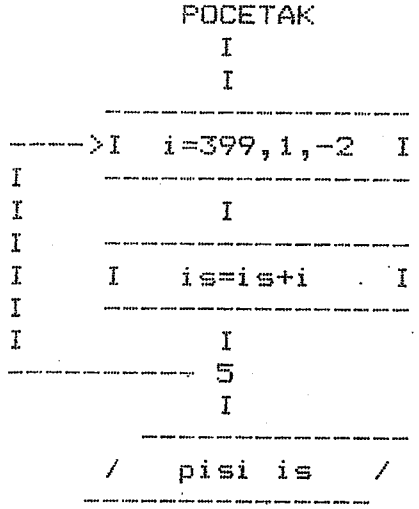
```
          DO 5 I=1,7
             ...
             ...
             ...
          I=I+K
             ...
             ...
             ...
      5   CONTINUE
```

U DO petlju nije dozvoljeno uci izvana kao npr:

```
     DO  5  I=41,31, -1
     ...
     ...
     ...
10 CONTINUE
     ...
     ...
     ...
 5 CONTINUE
     ...
     ...
     ...
   GO TO 10
```

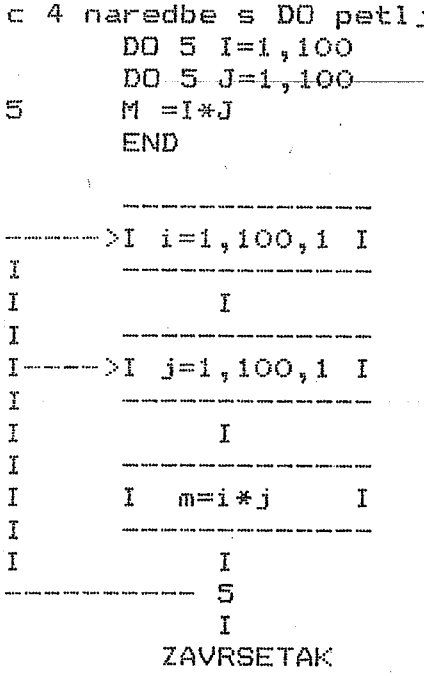
Iz DO petlje se smije izici. To je cesto i potrebno.

npr.1: Zbroj neparne brojeve od 400 do 1




```
      IS=0
      DO 5 I=399,1,-2
5     IS=IS+I
      PRINT *,'suma neparnih brj.=',IS
      END
```

npr.2: Naciniti tablicu umnozaka svih brojeva od 1 do 100.




```
c 4 naredbe s DO petljom
      DO 5 I=1,100
      DO 5 J=1,100
5     M =I*J
      END
```

```
c 16 naredbi bez DO petlje
      I=1
      J=1
5     M=I*J
      J=J+I
      IF(J.GT.100)THEN
         J=1
         I=I+1
         IF(I.GT.100)THEN
            GO TO 99
         ELSE
            GO TO 5
         END IF
      ELSE
         GO TO 5
      END IF
99    END
```

Uocimo da varijabla J unutrasnje DO petlje izmjenja vrijednost od 1 do 100 dok je vrijednost varijable I=1, nakon toga se poveca vrijednost I za 1 a varijabla J mjenja vrijednost od 1 do 100 i tako sve dok I ne poprimi vrijednost 101. Ocita je racionalnost pisanja programa za iterativne strukture upotrebom DO petlje.

5.8. Dodijeljivanje pocetnih vrijednosti promjenjivim

Naredba INTEGER, REAL, COMPLEX, DOUBLE PRECISION, LOGICAL i CHARACTER pored toga sto definiraju vrstu i duzinu promjenljivih mogu dodijeliti inicijalne vrijednosti promjenljivim.

npr.1: INTEGER SUM/101/,TOCKE(10)/10*1/

Varijabla SUM je tipa INTEGER s inicijalnom vrijednoscu 101, a varijabla TOCKA je INTEGER matrica od 10 clanova pri cemu su svi clanovi matrice dobili vrijednost 1.

npr.2: REAL A/1./,B(10)/1.1,8*0.,1.1/

Matrica B je tipa REAL do 10 clanova. Inicijalna vrijednost clanova matrice je: B(1)=1.1, B(2)=0, B(3)=0, B(4)=0, B(5)=0, B(6)=0, B(7)=0, B(8)=0, B(9)=0, B(10)=1.1

npr.3: COMPEX I11,C2*16(2)/(1.D-5,3.D5),(5D11,8.D80)/

Dvodimenzionalna matrica C2 je dvostruke preciznosti, pri cemu je :

C2(1)=1.D-5+3.D5i
C2(2)=5.D11+8.D80i

npr.4:
```
DOUBLE PRECISION A1B(3)/3*1.1D+0/
LOGICAL LE/'.TRUE.'/M/'.FALSE.'/
CHARACTER*8   CH,CHAR(10)*1
CHARACTER TI(2)*12/'programer','analiticar'/
```

TI je jednodimenzionalna literalna matrica od dva clana, ciji svaki clan moze biti najvise 12 karaktera a inicijalne vrijednosti su:

TI(1)='programer'
TI(2)='analiticar'

npr.5: Naredbom DIMENSION mogu se dodijeliti inicijalne vrijednosti tocno odredjenim clanova matrice.

DIMENSION F(0:4)/1.,2.,3.,4.,5./

Jednodimenzionalnoj matrici F od 5 clanova ciji je prvi clan sa indeksom 0 a posledjni sa indeksom 4 dodjeljene su vrijednosti:

F(0)=1., F(1)=2., F(2)=3., F(3)=4., F(4)=5.

5.9. DATA naredba

DATA naredba omogucuje inicijaliziranje varijabli, matrica i clanova matrica.

opci oblik:

```
DATA varijable/konstante/
        ili
DATA ((A(i,...),i=m1,m2,m3),...)/konstanta/
```

gdje je:

n je labela naredbe iza DO naredbe (a unutar programske jedinice) a koja je zadnja u sekvenci naredbi koje cè se visestruko izvoditi.

i je cijelobrojna, realna varijabla ili varijabla s dvostrukom preciznoscu.

m1,m2,m3 su konstante, varjable ili izrazi tipa INTEGER, REAL ili DOUBLE PRECISION (ne matrice), gdje je m1, pocetna vrijednost varijable i, m2 krajna vrijednost varijable i, m3 korak porasta varijable i. Ako m3 nije naveden podrazumjeva se da je m3=1, i ne smije biti m3=0.

Matrica se ucitava po kolonama ako je:

```
((A(I,J),J=1,Jmax),i=1,Imax)
```

a po redovima ako je:

```
((A(I,J),J=1,Imax),i=1,Jmax)
```

npr.1:

```
DIMENSION C(10)
DATA A,B/1.2,0.5/,C/10*1.2/
```

Konstante su poprimile vrijednosti A=1.2 i B=0.5. Svim clanovima matrice C je dodjeljena vrijednosti 1.2.

npr.2:

```
DIMENSION IA(10),LA(10)
DATA (IA(I),I=1,10)/1,2,3,7*4/,(LA(I),I=1,9,2)/5*3.3/
```

Neparnim clanovima matrice LA je dodijeljena vrijednost 3.3 a parnim clanovima nije dodijeljena vrijednost.

npr.3:

```
CHARACTER *4   A
DIMENSION A(10)
DATA A/'AB','CDE','FGHIJ'
```

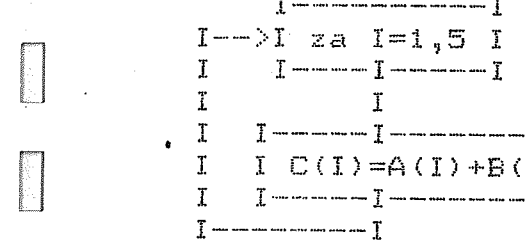
Prva tri clana literal matrice A imaju vrijednost:

A(1)='AB ', A(2)='CDE ', A(3)='FGHI'

pri cemu karakter J nije dodijeljen clanu matrice A(3) a clan matrice A(1) ima inicijalnu vrijednost blanko ("b") na poziciji treceg i cetvrtog karaktera.

ZADACI:

1. Zadana je matrica A=[1,2,3,4,5] i B=[6,7,8,9,0]. Napisati program za izracunavanje matrice C od pet clanova dane kao C=A+B.




```
      DIMENSION C(5)
      DIMENSION A(5)/1.,2.,3.,4.,5./,B(5)
      DATA (B(I),I=1,5)/6.,7.,8.,9.,0./
      DO 5 J=1,5
5     C(J)=A(J)+B(J)
      END
```

2. Od zadane dvodimenzionalne matrice A naciniti jednodimenzionalnu matricu B koristeci clanove drugog stupca.

```
          I 1 2 3 I
    A =   I 4 5 6 I
          I 7 8 9 I
```

```
      DIMENSION A(3,3),B(3)
      DATA((A(I,J),J=1,3),I=1,3)/1.,2.,3.,4.,5.,6.,7.,8.,9.,/
      DO 5 I=1,3
5     B(I)= A(I,2)
      END
```

3. Napisati program koji ce clanove glavne dijagonale matrice A(3,3) prepisati u matricu B(3).

```
        I7 8 9I
    A=I5 6 3I
        I1 4 2I
```

4. Napisati program koji ce zbrojiti matrice

```
        I2 8 I                  I8 2 I
    A=I3 7 I        i       B=I7 3 I
        I5 9 I                  I5 1 I
        I1 2 I                  I9 8 I
```

i rezultat smjesti u matricu C(4,2).

5. Napisati program koji ce u clan matrice M(i,j) smjestiti vrijednost i*j ako je i<j, odnosno i+j ako je i>=j.

6. Naci najveci i najmanji broj u nizu A(100) i ispisati ih.

6. ULAZNO / IZLAZNE NAREDBE

6.1. Opcenito

Ova grupa naredbi omogucuje prijenos podataka:
- iz memorije racunara na izlazni uredjaj
- u memoriju racunara sa ulaznog uredjaja
- u memoriju iz memorije racunara

Uredjaji za ulaz i izlaz podataka jesu npr.: tastatura, printer, ekran terminala, jedinica magnetske trake, busac i citac busenih kartica, busac i citac busenih papirnih traka, ploter, graficka stanica i drugi periferijski uredjaji.

Organizacija podataka, koji se prenose ulazno / izlaznim naredbama, u ovisnosti o tome kako se s njima rukuje i na kom dijelu sistema , dijeli se u tri tipa:

Sekvencijalna organizacija
Direktna organizacija
Unutrasnja organizacija

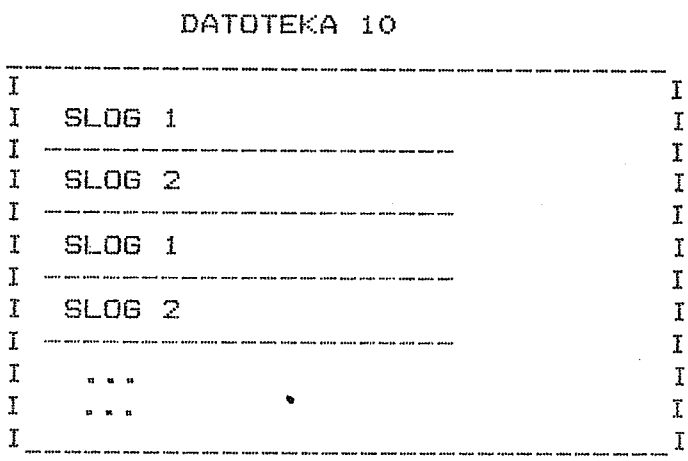
Naredbe za rad sa sekvencijalno organiziranim podacima jesu READ, WRITE, PRINT, BACKSPACE, ENDFILE, REWIND, OPEN, CLOSE, INQUIRE. Neke od naredbi se ne koristite kod nekih sekvencijalno organiziranih podataka ,npr.: naredbe BACKSPACES i REWIND kod citaca busenih kartica ne mogu biti koristene za rad a kod rada s jedinicama magnetskih traka mogu.

Naredbe za rad s direktno organiziranim podacima jesu READ, WRITE, OPEN, CLOSE, INQUIRE. Kod ovako organiziranih podataka direktno se pristupa podacima bez obzira na njihovu poziciju u skupu podataka.

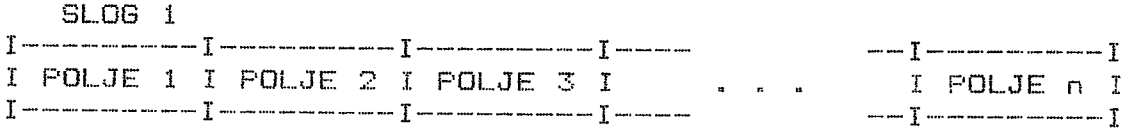
Za prijenos podataka iz intrne memorije u internu memoriju sluze naredbe READ i WRITE.

Nezavisni skupovi podataka koji se tretiraju kao nezavisne cjeline nazivaju se datoteke (FILES). Datoteka je skladiste podataka. Podaci se u datoteci organiziraju u slogove (RECORDS). Jedna datoteka moze imati vise razlicitih tipova slogova. Svaki tip sloga ima strukturu koja definira organizaciju podataka u slogu.

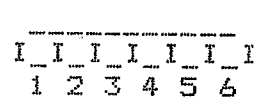
npr.1:



Datoteka sa imenom 10 se sastoji od datoteka organiziranih u dva tipa sloga SLOG 1 i SLOG 2. SLOG 1 sadrzi podatke koji su organizirani u polja. Struktura sloga je definirana rasporedom polja.



Svaki slog i svako polje se imenuje . Dobro je imenovati pojmove tako da ime govori o podacima koji se imenuju. Polje se sastoji od pozicija (mjesta).



POLJE 1 ima 6 pozicija, sto znaci da je duzina polja 6 mjesta i da se u POLJE 1 moze smjesti najvise 6 karaktera. Razlicita polja mogu imati razlicitu duzinu.

Neka se datoteka 10 sastoji samo od jednog sloga s imenom OTPNAP. Slog OTPNAP od dva polja OTPOR i NAPON. Polje OTPOR neka ima duljinu 5 pozicija a polje NAPON neka ima duljinu 6 pozicija. Datoteka 10 se moze graficki prikazati u obliku tablice koja u jednom slogu (redu) daje podatke o jednoj vrijednosti otpora i jednoj vrijednosti napona.

Datoteka 10.
slog OTPNAP

polje 1 OTPOR	polje 2 NAPON
17.3	325.6
18.5	315.8
19.8	306.7
22.3	250.3
137.8	000.0

Uocimo da sadrzaj datoteke 10 jesu podaci napisani organizirano. U cetvrtu pqziciju polja OTPOR i u petu poziciju polja NAPON se upisuje decimalna tocka.

Opcenito se u polja mogu upisati brojevi, slova ili bilo koja kombinacija karaktera. Naprimjer u polje za sifru nacrta upisuje se INTEGER broj a u polje za naziv nacrta CHARACTER tekst. Polja moze biti popunjeno ili ne sa podacima. Pozicija u polju moze i memora biti popunjena znakom. Naredbe za rad s podacima u datotekama ne operiraju s citavom datotekom odjednom vec s pojedinim slogovima.

Naredbe READ i WRITE omogucuju ulaz i izlaz slogova. Inicijalna vrijednost duljine sloga koja se podrazumijeva za UNIVAC 1100 je :

- za citac/busac kartica 80 pozicija
- za citac/busac papirne trake 80 pozicija
- za Sistem Data Format(SDF) datoteke 132 pozicije
- za ANSI datoteke 132 pozicije
- za printer 132 pozicije

Od dane duzine sloga moze se koristiti i manja ali ne veca. Naredbom OPEN mijenja se inicijalna duzina sloga datoteke na potrebnu (cesto vecu).

6.2. READ naredba

READ naredbom se prihvataju vrijednosti podataka i dodijeljuju u program.

Opci oblik:

READ(UNIT=u,FMT=f,ERR=s,END=sn,IOSTAT=ios) lista

ili krace

READ(u,f) lista

a moguc je i oblik

READ f,lista

ili samo

READ f

gdje je :

UNIT=u je oznaka broja datoteke. Rijec UNIT= nije obavezno pisati. Ako se ne pise tad se "u" mora pojaviti odmah nakon otvorene zagrade. Ne postoji standardna konvencija za numeriranje datoteka. Pojedini brojevi se dodijeljuju za periferne uredjaje kao npr.: citac kartica, printer, terminal itd. Domena dozvoljenih cijelih brojeva za UNIVAC 1100 se proizvoljno definira, na primjer (1,29) s tim da je:

5 standardni ulaz (npr. tastatura ili citac kartica)

6 standardni pristup (ekran terminala ili printer)

1 standardni busac

Ostali brojevi jesu imena datoteka na masovnoj memoriji. Primijetimo da ime datoteke ne moze biti karakter tipa, i da se UNIT=6 ne koristi u READ naredbi jer je taj broj rezerviran za WRITE naredbu. Oznaka broja jedinice za citanje i pisanje kod MICROSOFT FORTRAN 77 koji ide na IBM PC XT, IBM PC AT i druge kompatibilne racunare, je zvjezdicz("*").

FMT=f je oznaka labele naredbe FORMAT za formu ispisa podataka. Rijec FMT= nije potrebno pisati (kao i UNIT=).

f je labela naredbe FORMAT a moze biti:

1. numericka vrijednost
2. ime cijelobrojne varijabile cija je vrijednost jednaka broju labele naredbe FORMAT
3. karakter izraz, karakter varijabla, karakter matrica koji je jednak specifikaciji naredbe FORMAT npr. '(I5)'
4. zvjezdica "*" za slobodni format ispisa podataka.

U slucaju slobodnog formata podaci se na ulazu odvajanju zarezom, prazninom, kosom crtom ili novim slogom. Mogu se ucitati numericki, logicki ili karakter podaci.

ERR=s je clan koji omogucuje kontrolu ulaznih podataka. Cijeli clan ERR=s je moguce ispustiti a ako se koristiti obavezno ga je u punom obliku navesti. Ako je ovaj clan prisutan tad ce se u slucaju greske na ulaznim podacimsa kontrola izvrsenja programa prenijeti na naredbu s labelom "s". Ako clan nije prisutan tada ce se u slucaju greske na ulaznim podacima ispisati odgovarajuca poruka, koju je predvidio proizvodjac prevodioca (kompajlera) i nastaviti izvrsenje programa.

END=sn je clan koji omogucuje prepoznavanje kraja datoteke. Cijeli clan END=sn je dozvoljeno ispustiti a ako se koristi obavezno ga je u punom obliku navesti. Vrijednost "sn" odredjuje labelu naredbe (u istoj programskoj jedinici kao i READ naredba) na koju ce se prenijeti kontrola programa ako u toku citanja program naidje na kraj datoteke (znaci da je pokusano citanje sloga iza posljednjeg sloga u datoteci).

IOSTAT=ios omogucuje dodijeljivanje vrijednosti parametru "ios" u ovisnosti o tocnosti ucitavanja podataka. Cijeli clan IOSTAT=ios je moguce ispustiti, a ako se koristi obavezno ga je u punom obliku navesti. Pomocu ove naredbe se definira cijelobrojna varijabla ("ios") ili clan matrice, koja ce ako je prisutna kod izvrsenja naredbe primiti odredjenu vrijednost. Ta vrijednost je :

ios=0 ako je citanje zavrseno bez greske i nismo naisli na kraj datoteke

ios>0 ako je citanje zavrsilo u gresci i program nije naisao na kraj datoteke

ios<0 ako je kod pokusaja citanja pronadjen kraj datoteke.

Ovaj element ignorira prisustvo ERR i END elemenata i kontrola se prenosi u program. Sadrzaj varijable "ios" dan je u prirucniku proizvodjaca.

lista je popis imena polja koji zajedno cine ulazni slog a odvojeni su zarezom. Clanovi liste mogu biti:

1. imena varijabli
2. imena matrica
3. imena clanova matrica
4. karakter konstanta
5. implicitni DO popis.

Lista odredjuje kojoj ce varijabli biti dodijeljene vrijednosti podatka sa ulaznog sloga kad READ naredba zavsi. Sadrzaj prvog polja iz sloga dodijeljuje se prvoj vrjabili itd.

Primjeri:

a) READ(5,100) I,X,Y

Naredba omogucuje citanje podataka sa tastature po formatu ili obliku citanja koji je dan u naredbi FORMAT s labelom 100 a ucitane vrijednosti dodijelice se varijablama I, X i Y.

b) READ(15,101,ERR=99)IME,STAR

Naredba READ vrsi citanje podataka iz datoteke 15 i vrijednosti dodijeljuje varijablama IME i STAR. Duzina i tip polja (format citanja) dan je naredbom FORMAT koja ima labelu 101. U slucaju greske kod citanja, kontrola izvrsenja programa se prenosi na naredbu s labelom 99. Ako je polje STAR opisano kao INTEGER a u datoteci 15 je upisan podatak nekog drugog tipa javlja se greska kod citanja.

c) DIMENSION MAT(10),MR(5)
READ(5,100)A,B,MAT,MR(3)

Naredba READ ucitava varijable A i B matricu MAT i treci clan matrice MR sa tastature.

d) DIMENSION X(2,3),Y(2,3)
READ(10,200)(X(i,j),j=1,3),i=1,2)
READ(10,200)(Y(i,j),i=1,2),j=1,3)

Prvom naredbom READ se ucitava matrioca X red po red a drugom naredbom READ se ucitava matrica Y stupac po stupac.

e) READ(8,532,ERR=99,END=89)I,N,A,R

Ako se dodje do kraja datoteke 8, znaci da vise nema podataka u datoteci (predhodnim citanjem je procitan zadnji slog datoteke), kontrola daljnjeg rada programa se prenosi na naredbu s labelom 89.

6.3. WRITE naredba

WRITE naredbom se prenose podaci iz programa na periferne uredjaje (datoteke na magnetskim diskovima, terminal...)

opci oblik:

```
WRITE(UNIT=u,FMT=f,ERR=s,IOSTAT=ios)lista
                ili krace
            WRITE(u,f)lista
                ili
             WRITE f,lista
               ili samo
                WRITE f
                a moze
             PRINT f,lista
               ili samo
                PRINT f
```

Znacenje simbola dano je kod naredbe READ. Uocimo da slucaj f="*" daje oblik PRINT naredbe koju smo vec upoznali.
PRINT *,lista

UNIT=u je broj izlazene jedinice. Tumacenje parametara je isto kao i kod READ naredbe s tim da za "u" vrijedi:
u=6 standardni izlaz(npr. terminal ili printer)
u=5 standardni ulaz, a ne koristi se kod WRITE naredbe.

npr.:
a) WRITE(9,445,ERR=190)V1,V2,MAT
Izvrsenje ove naredbe upisuje sadrzaj varijabli V1 i V2 i matrice MAT u datoteku 9 koristeci naredbu FORMAT s labelom 445 za specifikaciju upisa. Ako se pojavi greska kod izvrsenja naredbe kontrola se prenosi na naredbu s labelom 190.

b) WRITE(FMT=445,UNIT=9,ERR=190)V1,V2,MAT
Naredba ima isti efekt kao i predhodna naredba.

c) PRINT 445,V1,V2,MAT
Ispisuje iste vrijednosti na printer ili ekran terminala.

6.4. FORMAT naredba

Naredba FORMAT specificira duzinu i tip za vrijednosti ulazno / izlazne liste i uredjuje podatke u slogu.

opci oblik:

f FORMAT(opisivaci)

gdje je:

f labela naredbe FORMAT, koja je uvijek cijelobrojna

FORMAT sluzbena rijec

opisivaci niz opisivaca ulazno / izlaznog sloga odvojenih zarezom koji opisuju polja. Ako ulazno / izlazna lista nije specifirana i naredba FORMAT nema opisivaca, jedan ulazni slog biti ce preskocen ili upisan prazan izlazni slog.

opis:

Ovo je opisna naredba i moze se pojaviti bilo gdje u programu. Ista FORMAT naredba moze sluziti vecem broju ulazno / izlaznih naredbi, u svim onim naredbama u kojima je navedena labela naredbe FORMAT. Naredbom FORMAT se opisuju polja na ulazno/izlaznim slogovima. Polje je bilo koji niz susjednih pozicija. Sirina polja u slogu je broj pozicija u polju. Na ulazu slog je podijeljen u polja odredjene sirine, u skladu sa specifikacijom u naredbi FORMAT, a na izlazu slog je sastavljen spajanjem izlaznih polja. Polje moze sadrzati neku vrijednost iz "lista" (vidi READ ili WRITE naredbu), moze u njega biti upisana literalna (karakter) konastnta ili moze biti preskoceno (upisana vrijednost praznoOg karaktera).

Navedeno je, kod READ i WRITE naredbe da parametar FTM=f moze biti, ne samo pozitivna INTEGER konstanta vec karakter matrica, karakter varijabla i karakter izraz. Ovo omogucuje da tokom izvrsenja programa mijenjamo parametre u naredbi. To nazivamo varijabilni format.

6.5. Opisivaci polja

Opisivaci polja sluze za odredjivanje karakteristika polja i mogu odrediti tip i sirinu polja, sadrzati karakter konstante ili broj preskocenih pozicija. Ako se koriste dva ili vise opisivaca u naredbi, oni moraju biti odvojeni jedan od drugog zarezom, dvotockom ili kosom crtom. Velicina INTEGER konstanti "w", "d", "p" i "e" koji se pojavljuju u daljnjem tekstu mora biti manja od 512.

Opisivaci polja jesu:

6.5.1. INTEGER opisivac

Opci oblik: nIw

I je oznaka INTEGER opisivaca. Polja zauzima "w" pozicija. Vrijednost INTEGER varijable iz "lista" se pojavljuje u polju pozicionirana desno. Ako je to izlazno polje i sirina polja nije dovoljna za ispis cijelobrojne varijable navedene u "lista" (bilo predznaka bilo cifri) tada se u polju pojavljuju zvjezdice i ukazuju na preusko polje. Ako je polje sire od potrebnog, tada ce se u polju lijevo od polja pojaviti praznine. U ulaznom polju vodece praznine se zanemaruju a nakon broja u polju se interpretiraju kao nule ako se koristi BN oblik polja. Ako u "lista" ulazno/izlazne naredbe stoje dvije ili vise INTEGER varijabli iste duzine uzastopno, tada umjesto ponavljanja opisivaca za svaku varijablu na mjesto parametra "n" upisujemo broj ponavljanja istog opisivaca, te "n" predstavlja broj uzastopnih jednakih polja.

Ako se zeli opisati tri uzastopna polja sirine pet pozicija umjesto da pise I5,I5,I5 pise se 3I5.

```
          I1 2 3 4 5I1 2 3 4 5I1 2 3 4 5I    slog
          I----------I----------I----------I
               I5         I5         I5        = 3I5
```

U jednom slogu mogu se pojaviti polja istog tipa razlicite duzine pa tad pisemo za:

```
          I1 2I1I1 2 3I     slog
          I---I-I-----I
           I2  I1  I3       =I2,I1,I3
```

npr.1: Ako je dan program:

```
      NAD=730000
      I=20
      WRITE(6,100)I,NAD
100   FORMAT(I3,I4)
      END
```

Rezultat rada programa je ispis varijabli I i NAD u izlazni slog: 20****
sto je:

```
                I        NAD
              1 2 3 4 5 1 2 3 4
              I I I I2I0I*I*I*I*I
```

npr.2: Ako je dan program:

```
      READ (5,101)K
101   FORMAT(I5)
      WRITE (6,101)K
      END
```

tad ce izlaz za ulaz biti:

```
      ulaz   izlaz
      ------------
      I  123I  123I
      I  17 I  170I
      I15   I15000I
      I     I    0I
      I 5234I 5234I
      I     I     I
```

6.5.2. Drugi oblik INTEGER opisivaca

Cijelobrojni opisivac se moze napisati u drugom obliku,

opci oblik : nIw.d

samo za naredbu WRITE. Ovdje "n","I" i "w" imaju isto znacenje kao kod opisivaca nIw. Parametar "d" odredjuje da ce u desno polju biti uvijek ispisano najmanje "d" cifara. Ukoliko je sirina "d" veca od broja cifara varijable koja se ispisuje, tad se praznine popunjavaju nulama. U slucaju ispisa predznaka, "w" mora biti veci od "d".

npr.1: Ako je dan program

```
      READ (5,102)K
102   FORMAT (I5)

      WRITE (6,103)K
103   FORMAT (I5.3)
      END
```

tad ce za dani ulaz izlaz biti:

ulaz	izlaz
I 1I	001I
I 378I	378I
I 21 I	210I
I-4325I	-4325I
I 27I	027I
I1 I10000I	

6.5.3. REAL opisivac s fiksnim zarezom

opci oblik: nFw.d

Oznaka opisivaca realnih varijabli je "F". Varijable u "lista" mogu biti REAL, COMLEX(jedna od para) i DOUBLE PRECISION. Polje zauzuma "w" pozicija na izlazu. Broj decimalnih pozicija u polju je "d" i smjestene su desno od decimalne tocke. Za negativne brojeve predznak minus stoji neposredno ispred cijelobrojnog dijela. Podrazumjeva se da je predznak pozitivan ako predznak nije naveden. Ako je izlazno polje nedovoljno siroko u polje se ispisuju zvijezdice. Sirina polja mora biti:

w>=2+d+c

jedno mjesto za predznak, jedno mjesto za decimalnu tocku, "d" mjesta za decimalni dio i "c" mjesta za cijeli dio realnog broja. Broj cijelih mjesta moze biti veci ili jednak nuli. Parametar "n" ima isto znacenje kao kod opisivaca nIw, a sluzi za ponavljanje istog opisivaca.

npr.1: Za ispis realnog broja 134,14 potrebna sirina polja je najmanje w=6, broj decimalnih mjesta d=2 i pisemo opisivac F6.2, za broj 2437,385 pisemo F8.3, a za broj 0,075 pisemo F4.3.

Na ulazu, opisivac Fw.d predstavja polje duljine "w" pozicija od kojih se ocekuje "d" pozicija za decimalne vrijednost. Ako se na ulazu pojavi broj sa decimalnom tockom, zanemaruje se vrijednost "d", i uzima se stvarni broj decimalnih pozicija ulazne vrijednosti ako je ona manja ili jednaka od "d". Ne moze se ucitati broj decimala veci od "d". Vodece prazne pozicije se ne uzimaju u obzir, a prazne pozicije iza broja, ali u polju, se u slucaju upisa decimalne tocke zanemaruju odnosno bez decimalne tocke pretvaraju u nule (kao kod opisivaca Iw).

npr.2: Za ucitati brojeve: 13,2; 1256,13; -17,231 treba koristiti opisivace F4.1; F7.2; F7.3 ili moze i F6.2; F8.2; F9.3. Smanjenjem broja "w" ne bi smo ucitali cijele pozicije a broj "d" decimalne pozicije.

6.5.4. REAL opisivac s pomicnim zarezom

opci oblik: nEw.d

Sluzi za ispis varijabli iz "lista" tipa REAL, COMPLEX (jedna od para) i DOUBLE PRECISION. Oznaka opisivaca je "E". Polje zauzima "w" pozicija. Vrijednost varijable u "lista" pojavljuje se kao decimalni broj u polju pozicion desno u obliku:

s.xxxxxseee

gdje je: "xxxxx" broj decimalnih mjesta "d" naveden u opisivacu (broj cijelih mjesta je uvjek jednak nuli), "s" je znak plus ili minus i zauzima jednu poziciju, i "eee" je eksponent broja 10 pisan u tri pozicije. Plus ili minus se upisuje u poziciju "s" ovisno o tome da li je eksponent "eee", odnosno predznak broja, pozitivan ili negativan. Ukoliko je broj pozitivan tada se "+" ne ispisuje na izlazu. Ukupni broj pozicija polja "w" dan je relacijom:

w=6+d

(1 mjesto za predznak broja, 1 mjesto za decimalnu tocku, 1 mjesto za predznak eksponenta, 3 mjesta za eksponent i "d" mjesta za brojevnu vrijednost).

Ako je polje sire nego sto je potrebno, broj se pozicionira desno a lijevo su vodece praznine. Ako polje nije dovoljno siroko, u polje se ispisuju zvijezdice.

Na ulazu, Ew.d opisivac se ponasa kao Fw.d opisivac.

npr.1: Ako je dan program

```
      READ(5,100)R
100   FORMAT (E10.3)
      WRITE(6,101)R
101   FORMAT(E15.3)
      END
```

onda ce za dani ulaz slijediti izlaz:

ulaz	izlaz
1.2	.120+001
1.2	.120-001
7243.123	.724+004
12345678.9	.123+008
-12	-.120+006
13.12E003	.132+005
13.12E003	.131+032
1.2E-5	.120****
-0.034E12	-.340+120

treba razlikovati "E" opisivac u FORMAT naredbi od slova "E" unutar REAL konstante koja se pojavljuje na ulaznom polju.

6.5.5. Specificni REAL opisivac s pomicnim zarezom

opci oblik: nEw.dEe

To je specijalni slucaj opisivaca nEw.d a sluzi za ispis brojeva s proizvoljnim brojem pozicija za eksponent. Broj ispisuje u obliku

s.xxxxxse...

Parametar "e" odredjuje broj pozicija za ispis eksponenta, a ako je taj broj manji od potrebnog u polje se ispisuju zvijezdice.

6.5.6. DOUBLE PRECISION opisivac s pomicnim zarezom

opci oblik: Dw.d

Opisivac je indentican opisivacu Ew.d a sluzi za ispis brojeva s dvostrukom preciznoscu. Tri pozicije su namjenjene za eksponent.

6.5.7. Tretiranje praznih pozicija

opci oblik: BN ili BZ

Ovaj opisivac sluzi za interpretiranje praznih pozicija iza broja kod unosa podataka. Ako unutar naredbe FORMAT pisemo kod BN onda se svi ucitani brojevi nakon njega citaju tako, da se prazne pozicije u polju nakon broja ne tretiraju kao nule, vec se ucitava samo cifarska vrijednost broja. To svojstvo vrijedi dok se ne zavrsi naredba FORMAT ili dok se ne upise kod BZ. Prazne pozicije u broju iza polja, kod ucitavanja brojevne vrijednosti se tretiraju inace kao nule.

npr.:

```
100   FORMAT(BN,I5,BZ,F3.1/BN,5I4/A5,3F8.2)
```

6.5.8. Ispis predznaka plus ("+")

opci oblik: SP ili SS

Ovaj kod u naredbi FORMAT sluzi za kontrolu ispisa predznaka plus. Ako se nalazi kod SP pozitivni brojevi ce se ispisati s predznakom plus. Kod SS ili samo S iskljucuje ispis predznaka plus. Predznak minus se uvijek ispisuje.

npr.:

```
100   FORMAT(I3,SP,5F4.1,E8.3/3E12.4E4)
```

6.5.9. Logicki opisivac

opci oblik: Lw

Sluzi za ucitavanje ili ispisivanje vrijednosti logickih varijabli. Ako je napisan opisivac Lw za ispis varijable iz "lista" cija je vrijednost .TRUE., tad ce u polju duzine "w" pozicija biti ispisano slovo "T" desno pozicionirano sa "w-1" praznom pozicijom. Ako je varijabla .FALSE. ispisuje se slovo "F" na isti nacin. Kod ucitavanja vrijednosti, pretrazuje se polje s lijeva na desno i trazi vrijednost "T" ili "F" i na osnovu njih varijabli dodijeli vrijednost .TRUE. ili .FALSE. . Svi karakteri osim "T" i "F" u ulaznom polju se ignoriraju. Ako se u polju ne pojavi slovo "T" ili "F" varijabli se ne dodijeljuje vrijednost i pojavljuje se upozorenje.

npr.:

```
      LOGICAL X,Y
      READ(5,100)X,Y
      FORMAT(L1,L2)
```

6.5.10. Alfanumericki opisivac

opci oblik: nAw ili samo A

Sluzi za ulaz / izlaz alfanumerickih varijabili. "A" je oznaka alfanumerickom opisivaca, "n" je broj istih uzastopnih alfanumerickih polja, a "w" je duzina polja u pozicijama. Neka je duzina karakter varijable "l" pozicija. Ako je "w" vece lil jednako od "l" tada ce se na izlazu u polju pojaviti lijevih "w" pozicija bez upozorenja da je desnih "w-l" pozicija neispisano. Ako je "w" manje od "l" tad se ucitaju iz polja lijevih "w" pozicija a u "l-w" pozicija varijable su praznine. Ako je "w" vece ili jednako od "l" tad se iz desnih "l" pozicija dodijeljuje vrijednost varijabli a ostale vrijednosti iz ulaznog polja se ignoriraju.
Ako je w=0 ni jedan se znak ne prenosi. Ako se ispusti upisati "w" tada se prenosi onoliko pozicija, koliko je duga karakter varijabla.

```
      CHARACTER*4 X, A*8
      READ(5,10)A,X
10    FORMAT(A3,A5)
      WRITE(6,20)X,X
20    FORMAT(A6,A3)
      END
```

onda za dati ulaz slijedi izlaz:

```
I----------I---------I
IABCDEFGHIJIDEFGH DEFI
I12345     I45    45 I
I      AA  I   AA    I
I     XCD  I  XCD   XI
I          I         I
```

npr.1: Ako je dan program:

```
      CHARACTER*4 T1,T2,T3,T4
      READ(5,100)T1,T2.T3,T4
100   FORMAT(A4,A2,A4,A3)
      END
```

Za ulaz:IME I PREZIME , varijable ce poprimiti vrijednosti:

T1 je "IME"
T2 je "I "
T3 je " PREZ"
T4 je "IME".

6.5.11. Hollerit opisivac

opci oblik: wHh1h2h3 ... hw

Sluzi za ispis karaktera. "H" oznaka Hollerit opisivaca, "w" je broj pozicija koji zelimo ispisati odnosno broj karaktera iza "H" opisivaca, "h1", "h2",...."hw" su karakteri iz tabele ASCII karaktera (vidi Dodatak A.) koje zelimo ispisati (mogu biti i preaznine ("b")). Ovaj opisivac nije u vezi sa izlaznon "listom" iz WRITE naredbe. Maksimalan dozvoljeni broj karaktera (najveca vrijednost od "w"), moze biti 511.

6.5.12. Literal opisivac

opci oblik: 'h1h2...hw'

Sluzi za ispis karaktera. "h1", "h2", ..."hw" je niz karaktera iz tabele ASCII karaktera (vidi Dodatak A.) Niz karaktera se navodi izmedju apostrofa. Efekat ovog opisivaca je identican Hollerith opisivacu. Ako se zeli ispisati apostrof, tada se na mjestu gdje dolazi apostrof u tekstu pisu uzastopno dva apostrofa. Literal i Hollerith opisivac se koriste za ispis naslova, tekstova i sl..

npr.1: Prikaz upotrebe Hollerith i Literal opisivaca u programu:

```
      CHARACTER B*14
      I=10500
      B='DANKO MARKOVIC'
      WRITE(10,101)B,I
101   FORMAT(14H IME I PREZIME,A15,' ZARADA =',I7)
      END
```

rezultat ce biti ispisan u datoteku 10 a imati ce oblik:

```
>IME I PREZIME DANKO MARKOVIC ZARADA =  10500
```

6.5.13. Opisivac za skok

opci oblik: wX

"X" je oznaka opisivac za skok, "w" je broj pozicija koje ce biti preskocene.

npr.1: Ako je dan program:

```
      II=455
      A=5.88
      WRITE(6,100)A,II
100   FORMAT(1X,F6.2,5X,I2)
      END
```

biti ce ispisan rezultat:

```
>  5.88     **
```

gdje "**" oznacavaju da je za ispis varijable "II" nedovoljno dvije pozicije.

6.5.14. Pozicioniranje na mjesto u slogu

opci oblik: Tw

"T" je oznaka opisivaca, "w" je pozicija na koju se zelimo pozicionirati. Moze se koristiti bilo na ulazu bilo na izlazu. Na izlazu je FORMAT(10X,A2) isto sto i FORMAT(T11,A2). Pozicija na izlaznom slogu od prve do desete biti ce ispunjene prazninama, a varijable ce poceti sa ispisom od jedanaeste pozicije. Opisivaci "T" ne moraju biti uredjeni od najmanjeg prema najvecem unutar sloga.

npr.1: Ako je dan program:

```
      WRITE(6,100)
100   FORMAT(T10,'ADRESA',T5,'IME',T20.'TELEFON')
      END
```

biti ce ispisan rezultat:

```
>     IME  ADRESA     TELEFON
```

Pored opisivaca polja postoje u naredbi FORMAT i druge konvencije za kontrolu citanja i pisanja.

6.5.15. Prelaz na novi slog

Prelaz u novi slog postize se upotrebom kose crte (razlomacke crte) "/". Vise uzastopnih kosih crta znaci preskok vise slogova. Moze se koristiti i na ulazu i na izlazu. Odvajanje zarezom ispred i iza kose crte nije potrebno.

npr.1: Napisite naredbu FORMAT za ispis integer varijable duljine 10 pozicija u cetvrtom redu petoj koloni.

```
100 FORMAT(///T5,I10)
```

npr.2: Primjer naredbe

```
100 FORMAT(' ',I5/' ','TEKST')
```

npr.3: Ako je dan program:

```
      READ(15,10)X
      READ(15,11)y
10    FORMAT(F6.1/)
11    FORMAT(F6.1/)
      END
```

biti ce procitani podaci iz prvog i trceg sloga datoteke 15

```
1. slog        > 123.3     - procitan
2. slog        >           - preskocen
3. slog        >1111.7     - procitan
4. slog        >
```

6.5.16. Kontrola stampe

Prvi znak ima utjecaj u izlaznom slogu na kontrolu stampe. Ako je rezultat rada upucen na stampanje, onda se na oblik stampe moze utjecati prvim znakom u izlaznom slogu.

prvi znak	znacenje
praznina("b")	obican prored
0	dvostruki prored
1	skok na prvi red nove stranice
+	ispis bez skokova u novi red

```
      WRITE(6,100)
100   FORMAT(1H1,          <--- skok na novu stranicu
100   FORMAT(' TEKST'      <--- normalni ispis
100   FORMAT('TEKST'       <--- ispisana bi bila rijec "EKST"
```

Ukoliko je izlazni slog upucen na stampanje onda se prvi znak ne moze koristiti za ispis karaktera.

6.5.17. Ponavljanje grupe opisivaca

Ako se niz opisivaca zeli ponoviti vise puta onda se ta grupa opisivaca stavlja u zagrade a ispred zagrada broj koji oznacava koliko se puta ponavlja ta grupa opisivaca.

```
npr.1: FORMAT(I4,F14.7,2(I4,F6.1))
                    kao da pise
       FORMAT(I4,F14.7,I4,F6.1,I4,F6.1)
```

Najveci broj koji moze stajati ispred zagrade je 512. Jedna grupa opisivaca u zagradama se moze pojaviti u najvise tri nivoa zagrada.
Ako je broj varijabli iz ulazno / izlazne "lista" veci od odgovarajuceg broja opisivaca naredbe FORMAT ,onda se za ulaz / izlaz preostalih varijabli ponovo iskoristavaju opisivaci od prve zagrade u naredbi FORMAT (ponavlja se citava naredba FORMAT). Broj varijabli u ulazno/izlaznoj "listi" moze biti manji od broja opisivaca u naredbi FORMAT i obrnuto.

Prazan format se naziva naredba FORMAT bez opisivaca kao:

```
FORMAT( )
```

koja preskace jedan slog na ulazu i ispisuje prazan slog na izlazu.

npr.2: Program ce ucitati 6 varijabli tipa INTEGER iz 6 ulaznih polja, a svako ulazno polje je duljine 3 pozicije.

```
    READ(5,10)I,J,K,L,M,N
10  FORMAT(6I3)
    END
```

Ulaz moze biti> 1 12 2 7125
pa ce varijable poprimiti redom vrijednost :1,12,2,7,125,0

```
npr.3: Upotreba cijelobrojnog opisivaca polja
       K=851
       WRITE(6,100)K
100    FORMAT(1X,I5)
       WRITE(6,200)K
200    FORMAT(1X,I5.4)
       WRITE(6,300)K
300    FORMAT(1X,I2)
       END
```

Cijelobrojna varijabla K biti ce ispisana po formatu 100 kao: > 851, po formatu 200 kao> 0851, po formatu 300 kao greska>** jer dvije pozicije odredjene naredbom FORMAT nisu dovoljne za ispis tri cifre varijable K.

```
      npr.4: Primjer za realni opisivac polja
      A=32.5
      WRITE(6,100)A
100   FORMAT(1X,F7.2)
      WRITE(6,200)A
200   FORMAT(1X,E10.2)
      WRITE(6,300)A
300   FORMAT(1X,D10.2)
      END
```

Realna varijabla A ce biti ispisana po formatu 100 kao > 32.50, a po formatu 200 kao > .33+002, a po formatu 300 kao > .33+002. Opisivac D je namjenjen za varijable tipa DOUBLE PRECISION.

npr.5: Primjer za karakter opisivac

```
      CHARACTER*4 X,Y*8
      READ(5,100)X,Y
100   FORMAT(A3,A5)
      WRITE(6,200)X,Y
200   FORMAT(1X,A6,A3)
      END
```

Ako se na ulazu unese tekst>ABCDEFGHIJKLMN biti ce X='ABC ' i Y='DEFGH ' nakon ucitavanja po formatu 100. Ispis po formatu 200 je > ABCDEF

npr.6: Primjer za ponavljanje grupe opisivaca. Dozvoljeno je ponavljanje grupe opisivaca u grupi koja se ponavlja, do ukupno najvise cetri nivoa.

```
      FORMAT(1H1,7(I5,2(I3,F4.2,10(A4,I1,8(I5,F7.2)))))
```

npr.7: Ako je parametar formata ime cijelobrojne varijable, onda je vrijednost varijable labela naredbe FORMAT.

```
      CHARACTER A*10,B*5
      ILAB=200
      READ(10,FMT=ILAB)A,B
200   FORMAT(A10,A5)
      END
```

npr.8: Ako je parametar formata karakter izraz, onda sam izraz odredjuje formu ispisa varijabli iz "lista".

```
      CHARACTER*7 A,B
      INTEGER C
      READ(10,'(2A7/1X,I3)')A,B,C
      END
```

npr.9: Ako je parametar formata karakter varijabla, onda je vrijednost karakter varijable format ispisa varijabli iz "lista".

```
      REAL BROJ
      CHARACTER VAR*8
      DATA VAR/'(F7.3)'/
      READ(10,VAR)BROJ
      END
```

npr.10. Ako je parametar formata karakter matrica, onda je vrijednost clanova matrice forma ispisa. Ako u programu mijenjamo sadrzaj clanova matrice, mijenjamo i format ispisa. Ovaj slucaj se naziva varijabilni format i koristi za ispis varijabli ciji format ispisa je promijenljiv.
Naradba SUBROUTINE, koja ce biti koristena u ovom primjeru je opisana u poglavlju Potprogrami. Primjer pokazuje upotrebu varijabilnog formata u potprogramu za crtanje.

```
      SUBROUTINE PLOT1(IND,IS,X,B,F,N,FRM)
      DIMENSION A(101),B(N),F(N),FRM(4),FMT(6)
      character*24   fmt1
      equivalence (fmt(1),fmt1)
      DATA FMT(5) /'',101'/,FMT(6) /'A1)'/
  101 FORMAT(23X,101A1)
      DATA BLANK/' '/,AXIS/'I'/,PLUS/'+'/,STAR/'*'/,ZERO/'0'/
      DO 11 I=1,4
   11 FMT(I)=FRM(I)
      A(101)=AXIS
      IF(IND.EQ.1)A(101)=PLUS
      IF(IS.EQ.1) GO TO 2
      DO 1 I=2,100
    1 A(I)=BLANK
      A(1) =AXIS
      IF(IND.EQ.1) A(1)=PLUS
      GO TO 6
    2 IF (IND.EQ.1) GO TO 4
      DO 3 I=1,91,10
      A(I)=AXIS
      DO 3 J=1,9
    3 A(I+J)=STAR
      GO TO 6
    4 DO 5 I=1,91,10
      A(I)=PLUS
      DO 5 J=1,9
    5 A(I+J)=BLANK
    6 DO 7 I=1,N
      J=F(I)+1.5
      IF((J.LT.1).OR.(J.GT.101)) GO TO 7
      IF(A(J).LT.BLANK.OR.A(J).GT.BLANK)  GOTO9
      A(J)=B(I)
      GO TO 7
    9 A(J)=ZERO
    7 CONTINUE
      IF(IND.NE.1) GO TO 10
      WRITE(*,101) (A(I),I=1,101)
      RETURN
   10 WRITE(*,FMT1) X, (A(I),I=1,101)
      RETURN
      END
```

npr.11: Slobodan format rasterecuje korisnika programa da pamti format ulaznih podataka, sto inace zadaje poteskoce i dovoljno je ulazne podatke razdvojiti prazninom, zarezom, kosom crtom ili upisom svakog sljedeceg podatka u novi slog. Za program:

```
READ(5,*)A,B,C,D,...
WRITE(6,*)A,B,C,D,...
END
```

dovoljno je podatke, naprimjer numericke, napisati:

```
>101 236.87 0.00003,11,123.45/12.3 ...
```

6.5.18. Zadaci:

1. Napisati kakav bi bio ispis na stanpacu danog programa.

```
10    FORMAT('1 NOVA STRANICA',
     +/'+                    ..... isti red '.,
     +/'0    novi red s proredom'
     +/'    novi red bez proreda')
      WRITE(0,10)
      END
```

izlaz bi imao izgled:

```
   > NOVA STRANICA    ..... isti red
   >
   >    novi red s proredom
   >   novi red bez proreda
```

2. Napisati program za ucitavanje dvije stranice trokuta i kuta medju njima a po kosinusovom poucku izracunati trcu stranicu. Na izlazu neka pise:

a= ... b= ... kut= ... c= ...

tako da umjesto tockica budu ispisane stvarne vrijednosti. Program neka zavrsi kad za jednu stranicu trokuta upisemo 9999, a do tad neka ide ponovo na ucitavanje stranica i kuta.

3. Za sve vrijednosti argumenta iz domene $-7,3<=X<=15,5$ izracunati vrijednost funkcije zadane:

$$F(X)=\begin{cases} 1-e^{\sin(\sqrt{X^2+1})} & \text{za } X<0 \\ 0 & \text{za } X=0 \\ \cos X & \text{za } X>0 \end{cases}$$

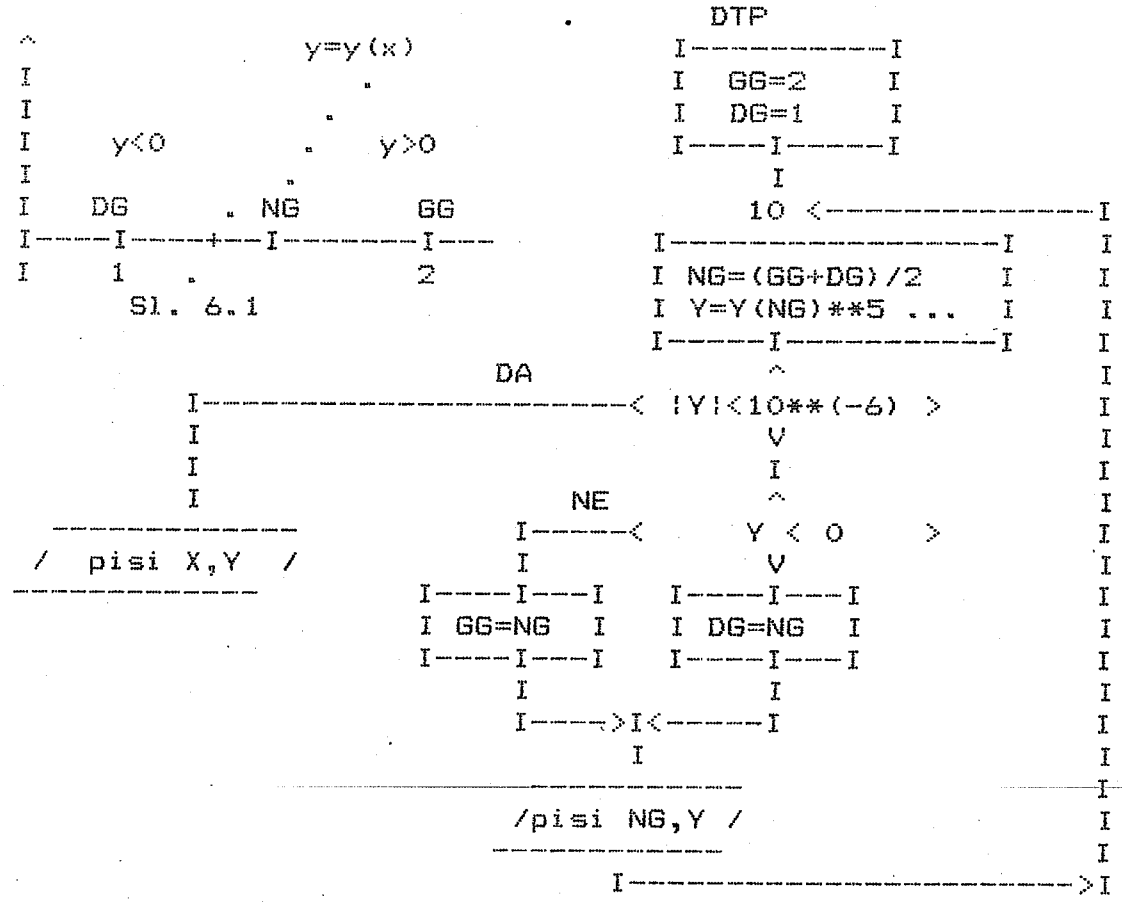
i ispisati X i F(X). Promjenu argumenta racunati s korakom 0,1.

4. Napisati program za racunanje srednje vrijednosti niza brojeva, tako da u prvom slogu ucitamo koliko brojeva je u nizu, a u ostalim slogovima brojeve iz niza. U slogu ima pet brojeva. Svaki broj je duzine 10 pozicija s dva decimalna mjesta. (Rjesenje za petlju: DO 5 BR=1,N+4,5)

5. Ako je zadan strogo rastuci polinom:

$$Y=X^5+2,801X^4-2,01X^3-2,998X^2-2,18X-6,028$$

a znamo da je nultocka u intervalu od 1. do 2. treba pronaci nultocku tako da tocnost bude manja od 1/1000000. Ispisati velicine Y i X u svakoj interaciji.




```
C  ZNACENJE SIMBOLA
C  NG - NOVA GRANICA, VRIJEDNOST ARGUMENTA
C  GG - GORNJA GRANICA
C  DG - DONJA GRANICA
C  Y  - VRIJEDNOST FUNKCIJE
C*****************************************************************
C
      REAL NG,GG,DG,Y
      GG=2.
      DG=1.
 10   NG=(GG+DG)/2.
      Y=NG**5+2.802*NG**4-2.01*NG**3-2.998*NG**2-2.18*NG-6.028
      IF(ABS(Y).GE.1.E-6 ) THEN
         IF(Y.LT.0.) THEN
              DG=NG

         ELSE
              DG=NG
         END IF
         WRITE(6,100) NG,Y
100      FORMAT (2X,F10.3,2X,F10.3)
         GO TO 10
       ELSE
          WRITE(6,101)I,NG,Y
       END IF
101    FORMAT(10X,' NUL TOCKA NADJENA U :',
     +I5,' ITERACIJA'/
     +5X,'X=',F10.6,2X,'Y=',F10.6)
       END
```

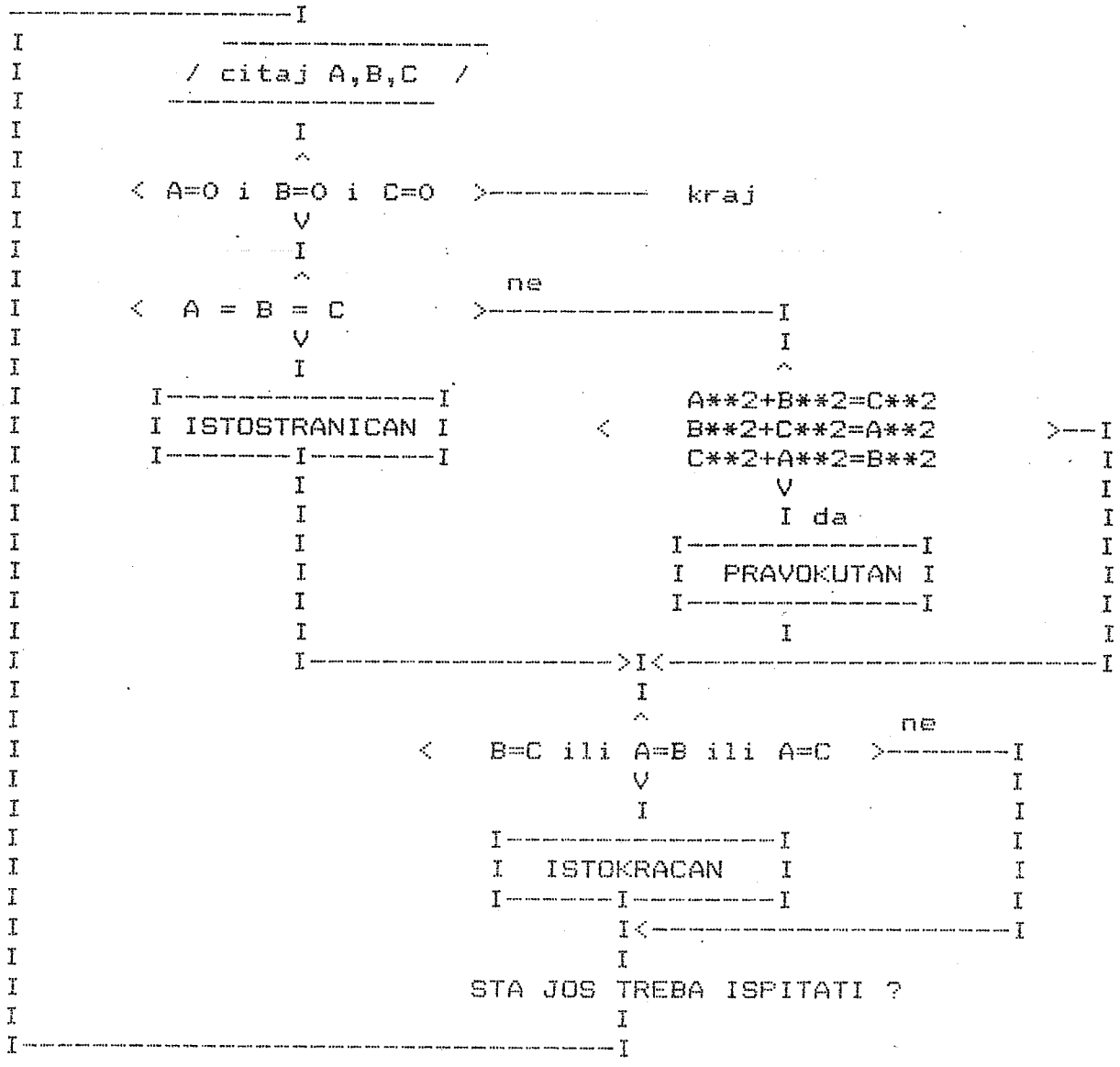
6. Ucitati tri realna broja A, B i C duzine 10 pozicija sa 3 decimale i uzeti da su to stranice trokuta. Ispitati i ispisati koje je sve vrste trokut:

- istostranicni
- pravokutan
- istokracan
- nemoguc

Ako su sve tri stranice jednake nuli prekinuti rad programa, u protivnom ici na ispitivanje i ucitavanje novih stranica trokuta.

Preporuke za izgradnju algoritma:

- nemoguc trokut eliminirati na pocetaku
- ispitati najvecu stranicu i pojednostaviti uvjete
- ako je istostranican, trokut je i istokracan ali nije pravokutan
- kontrolirati gresku kod ucitavanja stranica
- iskljuciti negativne stranice
- trokut je nemoguc ako je: A>B i A>C i A>B+C
- trokut je pravokutan ako je : A**2 = B**2 + C**2
- trokut je istostranican ako je: A=B=C
- trokut je istikracan ako je: A=B
- ispisati sve kombinacije istokracnosti i pravokutnosti
- moze li trokut biti nijedne navedene vrste?



7. Dan je program:

```
      LOGICAL A,B
      A=.TRUE.
      READ(5,100)B
100   FORMAT(L5)
      WRITE (6,110)A,B
110   FORMAT(1X,'A=',L3,' B=',L2)
      END
```

Vrijednost logickim varijablama sa tastature se dodijeljuje upisom u prvu kolonu slova "T" za TRUE ili "F" za FALSE.

Ako na ulazu pise > F na izlazu ce desno pozicionirano u polju ispisati slova "T" i "F" kao: >A= T B= F

Logicki opisivac "L" koristen u naredbi FORMAT sluzi za ulaz / izlaz logickih varijabli. Duzina polja je proizvoljna. Odredi izlaz ako bi na ulazu dodijelili vrijednost "T" varijabli "B" i ako sve logicke opisivace u programu zamjenimo opisivacem "L5".

8. Ako je dan program:

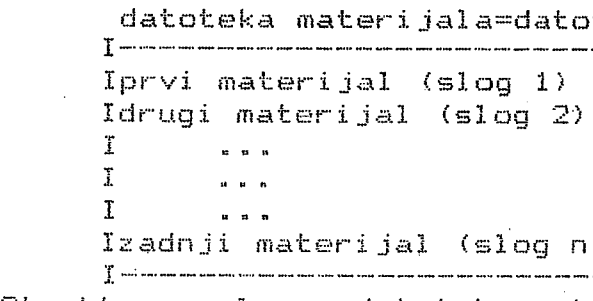
```
      A=.TRUE.
      B=.NOT.A
      C=.NOT.B
      WRITE(6,200)A,B,C
200   FORMAT(1H1,1X,2L4,L3)
      END
```

napisati kako bi izgledao ispis na terminal.

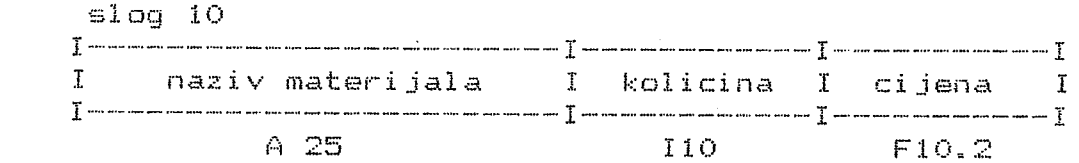
6.6. Datoteka

Datoteka je definirana u 8.1 kao skladiste podataka. Datoteka se sastoji od slogova, slogovi od polja a polje od pozicija. U jednu poziciju moze biti upisan jedan simbol iz tablice ASCII simbola (vidi DODATAK A.). U polja se upisuju podaci, kao npr., u polje "IME" se upisuju imena: MARKO, VLADO, IVANA, i sl., a u polje "NAPON" se upisuju numericke vrijednosti: 124.37, 110732.00, 0.327 (vidi poglavlje 6.1).
Sekvencijalno organizirana datoteka je skladiste podataka, sastavljena od niza podataka organiziranih u slogove, a slogovi podataka slijede jedan za drugim i pojedini slog se upisuje izvrsenjem jedne naredbe za upis, a cita se izvrsenjem jedne naredbe za pristup i to tako da se do nekog sloga moze doci samo iduci redom od jednog do drugog sloga. Rad sa sekvencijalnim datotekama prikazati ce mo na jednom primjeru.

Neka je datoteka logicki organizirana samo od jednog tipa sloga. Jedna datoteka nosi podatke o materijalima od kojih je sastavljen jedan proizvod. Posto su u datoteci podaci o materijalu proizvoda nazivasmo je datoteka materijala. Jedan materijal od koga je sacinjen proizvod odgovara jednom slogu datoteke. Svaki materijal ima naziv materijal. Ostali podaci o materijalu, trenutno od interesa su jedinicna cijena materijala i kolicina jedne vrste materijala ugradjeneu proizvod. Struktura sloga datoteke materijala sastoji se iz tri polja: naziva, kolicine i cijene. Svakom od pojedinih polja odredjuje se duzina u pozicijama. Neka je polje za naziv dugo 25 pozicija, polje za kolicinu 10 pozicija i polje za cijenu 10 pozicija. Pored duzine polja odredjuje se i tip polja. Polje naziv je alfanumericko i njega se mogu upisati bilo koji niz ASCII karaktera. Polje kolicine je cijelobrojno i uzimamo da su kolicine iskljuceno cijeli brojevi. Polje cijena je realno i interesuje nas cijena s tocnoscu od dva decimalna mjesta. Odgovarajuci FORTRAN opisivac za polje: Naziv, kolicina i cijena su:A25,I10iF10.2. Ove opisivace koristimo u prpogramu u naredbi FORMAT. Fortran-program moze pristupati datotekama po njihovim numerickim imenima, pa neka je datoteka materijala oznacena brojem "10". Datoteka 10 graficki mozemo predstaviti:



Strukturu sloga datoteke materijala (slog 10) mozemo graficki predstaviti:

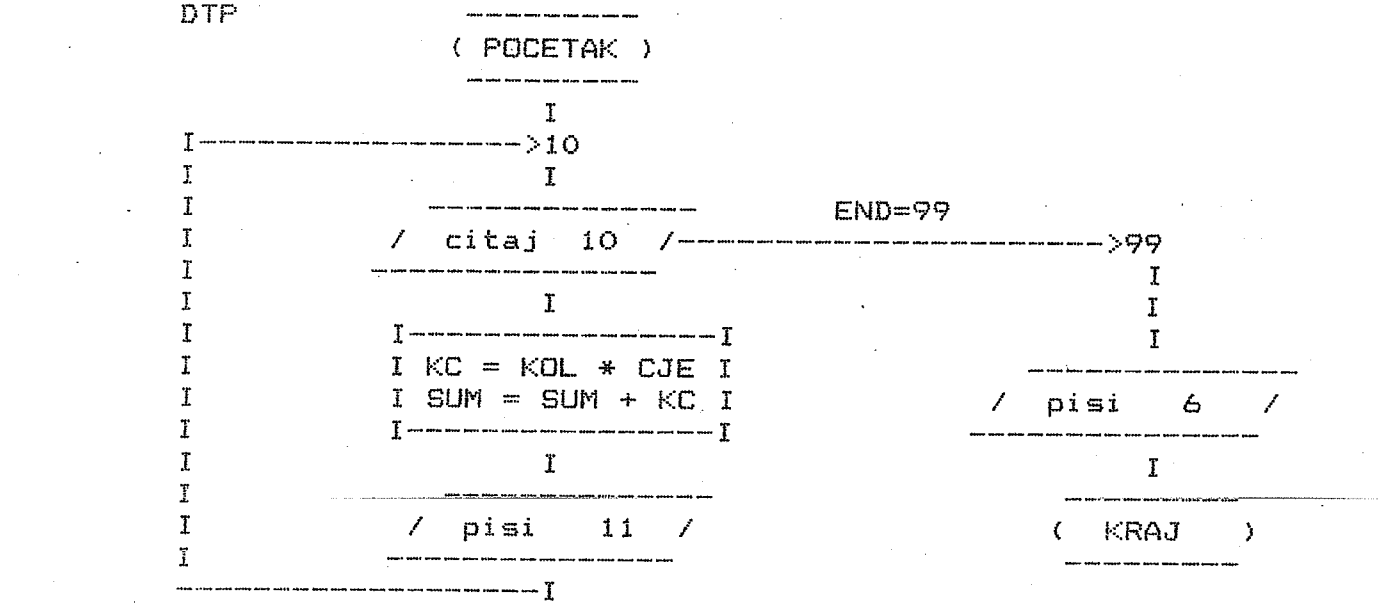


Zadatak je: napisati program za citanje datoteke materijala i izracunati produkt kolicine i jedinicne cijene za svaki materijal i u datoteku ukupne cijene materijala (oznacene kao datoteka 11) upisati podatke s logickom struktukturom sloga:

slog 11

naziv materijala	kolicina	cijena	kolicina * cijena
A25	I10	F10.2	F12.2

Na kraju rada ispisati na ekran poruku "ukupna cijena svih materijala je:" i ispisati ukupnu cijenu. Ukupna cijena je suma svih vrijednosti u polju kolicina * cijena.



Program:

```
C  DEFINIRANJE TIPA VARIJABLI I INICIJALIZACIJA
C*****************************************************************
      CHARACTER *25 NAZIV
      INTEGER KOL
      REAL CJE,KOLCJE,KC,SUM/0/
C*****************************************************************
C  CITANJE JEDNOG SLOGA IZ DATOTEKE 10
C******
10    READ(10,100,ERR=98,END=99)NAZIV,KOL,CJE
100   FORMAT(A25,I10,F10.2)
C*****************************************************************
C  RACUNANJE KOEFICIJENTA I UKUPNE SUME
C******
      KC=KOL*CJE
      SUM=SUM+KC
```

```
C*****************************************************************
C  UPIS PODATAKA U DATOTEKU 11
C******
      WRITE(11,110)NAZIV,KOL,CJE,KOLCJE
100   FORMAT(A25,I10,F10.2,F12.2)
      GO TO ,10
C*****************************************************************
C  GRESKA KOD CITANJA DATOTEKE 10
C    POGRESAN SLOG NE UPISUJEMO U DATOTEKU, VEC NA TERMINAL
C    NASTAVLJAMO CITANJE DAT. 10 JER MOZDA JOS IMA POGRESNIH
C******
98    PRINT *,' GRESKA KOD CITANJA DAT.-10'
      WRITE(6,100)NAZIV,KOL,CJE
      GO TO 10
C*****************************************************************
C  NEMA VISE SLOGOVA U DATOTECI 10 PA ISPISUJEMO UKUPNU SUMU
C******
99    WRITE(6,120)SUM
120   FORMAT(' UKUPNA CIJENA SVIH MATERIJALA JE:',F14.2)
      END
```

6.7. BACKSPACE naredba

Nakon citanja sloga u datoteci, pokazivac sloga se pozicionira na prvi sledeci slog i on ce biti citan slijedecom READ naredbom. Ako zelimo pokazivac vratiti na predhodni slog i procitati vec procitan slog i opcenito ici unazad po datoteci, koristimo naredbu BACKSPACE.

opci oblik:

```
BACKSPACE( UNIT=u,ERR=s,IOSTAT=ios)
        ili krace :
BACKSPACE(u,ERR=s)
        ili samo :
BACKSPACE u
```

Znacenje parametra UNIT, u, ERR, s, IOSTAT, ios je opisano u 6.2. READ naredba.

Jedno izvrsenje naredbe "BACKSPACES u" pozicionira pokazivac za pristup jedan slog unazad u datoteci "u".

Ako je pokazivac pozicioniran na prvi slog tad izvrsenje naredbe BACKSPACES u nema ucinka.

npr.:

```
READ(10,100) lista
READ(10,100) lista1
BACKSPACE 10
READ(10,200) lista2
  ...
  ...
  ...
```

prva naredba READ cita prvi slog u datoteci 10,
druga ----------drugi----------
treca ----------drugi----------.

6.8. ENDFILE naredba

Naredba ENDFILE stavlja oznaku kraja datoteke.

opci oblik:

```
.ENDFILE(UNIT=u,ERR=s,IOSTAT=ios)
                ili
 ENDFILE u
```

Znacenje parametra je opisano u 6.2. READ naredba. Jedno izvrsenje naredbe ENDFILE u upisuje oznaku kraja u datoteci u.

Primjer dan u 6.6. Datoteka, nadopunjen ovom naredbom, imao bi promjenu u dijelu programa nakon zadnjeg citanja datoteke 10, i to postavljanje "EOF" markice (oznaka kraja) u datoteku 11.

```
 99     ENDFILE 11
        WRITE(6,120)SUM
120     FORMAT(' UKUPNA CIJENA SVIH MATERIJALA JE:',F14.2)
```

6.9. REWIND naredbe

Naredbe REWIND pozicionira pokazivac za pristup na pocetnu poziciju (prvi slog u datoteci).

opci oblik:

```
REWIND(UNIT=u,ERR=s,IOSTAT=ios)
              ili
        REWIND u
```

Znacenje parametra je opisano u 6.2. READ naredba.

Izvrsenje naredbe REWIND u pozicionira pokazivac za pristup slogovima na prvi slog, bez obzira gdje se pokazivac trenutno nalazi.

npr.: Nakon uparivanja (jednakosti dvije varijable) varijable i polja datoteke umanjiti vrijednost varijable i zapoceti uparivanje od prvog sloga datoteke.

```
      I=236
10    READ(15,100,END=99)J
100   FORMAT(I3)
      IF(I.EQ.J)THEN
          REWIND 15
          I=I-1
      ELSE
          GO TO 10
      END IF
99    CONTINUE
      END
```

6.10. OPEN naredba

Pomocu OPEN naredbe se eksplicitno definiraju karakteristike datoteka koje se koriste u programu. Datoteke mogu biti sekvencijalne ili direktne. Za sekvencijalne i direktne datoteke.

opci oblik je:

```
OPEN( UNIT=u,IOSTAT=ios,ERR=s,STATUS=sta,ACCESS=acc,RECL=rl,
FORM=fm)
```

Parametar u od UNIT=u je obavezno pisati. Ostali parametri se koriste po potrebi. Za direktnu datoteku parametar "u" i RECL=rl je obavezno pisati.

Znacenje parametra : UNIT=u, IOSTAT=ios, ERR=s je opisano u 6.2. READ naredba.

STATUS=sta Parametar "sta" je karakter izraz koji moze poprimiti jednu od vrijednosti : OLD,NEW, SCRATCH. Ako je koristena vrijednost OLD obavezno je da datoteka postoji u memoriji racunaru. Ako je koristena vrijednost NEW, otvara se nova datoteka pod nazivom "u". Ako je koristena vrijednost SCRATCH, otvara se privremena datoteka, koja postoji dok se ne zavrsi program ili izvrsi naredba CLOSE. Ako je STATUS ispusten racunar ispituje da li datoteka postoji i postupa kao da je opcija OLD a ako ne postoji stvara se kao s opcijom SCRATCH.

ACCESS=acc Parametar "acc" je karakter izraz i moze poprimiti vrijednosti: SEQVENTIAL ili DIRECT. Ako ACCESS nije naveden podrazumijeva se DIRECT (vidi 6.12. i 6.13.).

RECL=rl "rl" je cijelobrojni izraz cija vrijednost mora biti pozivana. Vrijednost tog izraza odredjuje duzinu sloga u direktnoj datoteci racunatoj u pozicijama ako je FORM=FORMATTED i racunatoj u rijecima ako je FORM=UNFORMATTED.

FORM=fm "fm" je LITERAL izraz cija vrijednost je FORMATTED ili UNFORMATED. Ako je koristeno FORMATTED kod RECL se racuna duzina u pozicijama , a obrnuto se racuna u rijecima (npr. jedna ASCII rijec na SPERRY 1100 ima cetri pozicije). Ako parametar nije naveden podrazumijeva se UNFORMATED.

npr.: Otvoriti direktnu datoteku sa slogom dugim 400 pozicija.

```
OPEN(10,ERR=999,STATUS=NEW,ACESS=DIRECT,RECL=400,
+FORM=FORMATTED)
```

6.11. CLOSE naredba

Naredba CLOSE zatvara datoteku otvorenu OPEN naredbom.

opci oblik:

```
CLOSE(UNIT=u,IOSTAT=ios,ERR=s,STATUS=sta)
                    ili
                CLOSE u
```

STATUS=sta "sta" je karakter izraz koji moze poprimiti vrijednost KEEP ili DELETE. KEEP parametar zadrzava datoteku nakon rada programa na racúnaru a DELETE parametar brise datoteku pri izvrsenju naredbe CLOUS. Parametar KEEP ne moze biti koristen ako je kod OPEN naredbe STATUS='SCRATCH'. Ako je STATUS ispusten podrazumjeva se KEEP izuzev ako je kod OPEN status SCREATCH tad je status DELETE.

npr.:

```
CLOSE(12,STATUS='DELETE')
```

Neki prevodioci sami otvaraju i zatvaraju datoteke kojima pristupa program, bez obzira da li je to programer naveo ili ne, te olaksavaju pisanje programa, ali pravilno je svaku datoteku kojoj program pristupa otvoriti na pocetku programa i zatvoriti na kraju. Na taj nacin se olaksava citanje i odrzavanje programa, jer su datoteke popisane i eksplicitno definirana njihova svojstva.

6.12. Naredba READ za direktni pristup

Ako je datoteka organizirana kao datoteka s dirketnim pristupom, sto znaci da bilo kojem slogu datoteke mozemo pristupiti bez obzira gdje se nalazi pokazivac za pristup, koristimo grupu ulazno/izlaznih naredbi za direktan pristup. Naredba OPEN kod ACCESS='DIRECT' odredjuje se da je datoteka direktna, prije upotrebe naredbi iz ulazno/izlazne grupe. Za citanje podataka iz datoteke s direknim pristupom koristi se naredba READ ciji je

opci oblik:

```
READ(UNIT=u,FTM=f,REC=r,ERR=s,IOSTAT=ios) lista
                        ili
              READ(u,REC=r) lista
```

Parametri UNIT=u,FTM=f,ERR=s,IOSTAT=ios su opisani u poglavlju 6.2. READ naredba. Parametar "u" mora biti isti broj datoteke kao u naredbi OPEN.

Parametar REC=r je obavezan, pri tom "REC=" je sluzbena rijec a "r" je cijelobrojni izraz koji predstavlja relativnu poziciju sloga u datoteci. Izvrsenje READ naredbe dodijeljuje varijablama iz "lista" vrijednosti smjestene u r-tom slogu datoteke "u".

npr.:

```
OPEN(UNIT=10,RECL=250,ACCESS='DIRECT',FORM='FORMATED')
                    ...
                    ...
                    ...
READ(FMT=100,REC=37,UNIT=10,ERR=99)VAR1,MAT1
                    ...
                    ...
                    ...
```

Datoteka 10 je direktna i ima duzinu sloga do najvise 250 pozicija. Izvrsenjem naredbe READ dodijeljuje se sadrzaj tridesetsedmog sloga datoteke 10 varijablama VAR1 i MAT1 u skladu sa naredbom FORMAT s labelom 100.

6.13. Naredba WRITE za direktan pristup

Za upisivanje podataka u datoteku s direktnim pristupom koristi se naredba WRITE ciji je

opci oblik:

```
WRITE(UNIT=u,FTM=f,REC=r,ERR=s,IOSTAT=ios)lista
                    ili krace
              WRITE(u,REC=r)lista
```

Parametar REC=r je opisan u poglavlju 6.12. a ostali u 6.2.
U r-ti slog datoteke "u" upisati ce se vrijednosti dodijeljene varijablama iz "lista" pri jednom izvrsenju naredbe WRITE. Datoteka "u" mora biti otvorena naredbom OPEN. Ne moze biti upisan slog duzi od RECL=rl iz OPEN naredbe. Ako je upisan kraci slog od "rl" preostali dio sloga se popuni praznina. Ako se koristi u OPEN naredbi FORM=FORMATTED u protivnom je ostatak sloga nedefiniran. Ako je kod OPEN naredbe specifiran parametar FORM=FORMATTED obavezno je kod WRITE naredbe upotrebiti FMT=f.

npr.:

```
      OPEN(UNIT=15,RECL=12,ACCESS='DIRECT')
                        ...
                        ...
                        ...
   WRITE(REC=1,ERR=140,UNIT=15)A,B
                        ...
                        ...
                        ...
```

U datoteku 15 u prvi slog ce biti upisan sadrzaj A i B bez kontrole formata upisa.

6.14. Naredba READ za unutrasnje citanje

Omogucuje prenos i konverziju podataka iz unutrasnje memorije racunara u unutrasnju memoriju.

opci oblik:

```
READ(UNIT=iu,FTM=f,ERR=s,END=sn,IOSTAT=ios)lista
                    ili krace
              READ(iu,f)lista
```

gdje je:
UNIT=iu oderedjuje mjesto u centralnoj memoriji racunara odakle se prenose podaci u "lista". "iu" je karakter varijabla, karakter matrica ili njen element ili dio karakter varijable. Parametar "iu" je obavezno navesti.
Ostali parametri su opisani u poglavlju 6.2.

Naredba READ cita podatke tipa CHARACTER dane u "iu" i prepisuje ih u "lista" u skladu s naredbom FORMAT. Tip varilabli iz lista moze biti bilo koji i na taj nacin se vrsi konverzija podataka. Ispisuju se podaci iz "iu" sve dok se i posljednji ne prepise u "lista".

npr.1

```
      CHARACTER A*5,V1*2
      A='AB123'
      PRINT *,' A=',A
      READ(A,100)V1,I
100   FORMAT(A2,I3)
      PRINT *,' V1=',V1,'I='I
      END
```

npr.2

```
      CHARACTER A*1(5)
      DIMENSION I(5)
      A(1)='1'
      A(2)='2'
      A(3)='3'
      A(4)='4'
      A(5)='5'
      READ(A,100)I
100   FORMAT(I1)
      WRITE(6,200)(I(J),J=1,5)
200   FORMAT(1X,I1)
      END
```

Ako je "iu" varijabla, clan matrice ili dio varijable tad izvrsenje naredbe READ odgovara citanju jedenog sloga u listu, a ako je "iu" matrica onda je svaki clan matrice jedan slog.

Konverzija podataka tip-a CHARACTER je moguca u bilo koji tip.

6.15. Naredba WRITE za unutrasnje pisanje

Omogucuje prijenos i konverziju podataka iz unutrasnje memorije racunara u unutrasnju memoriju.

Opci oblik:

```
WRITE(UNIT=iu,FTM=f,ERR=s,IOSTAT=ios)lista
```

ili krace

```
WRITE(iu,f)lista
```

Parametar UNIT=iu je opisan u poglavlju 6.12 a ostali u poglavlju 6.2.

Varijable, matrice i clanove matrica raznih tipova navedene u "lista" prepisuje naredba WRITE u karakter varijablu, matricu ili clan matrice u skladu s naredbom FORMAT. Podaci se prepisuju u "iu" sve dok se ne ispisu svi iz "lista".

Konverzija podataka bilo kojeg tip-a u CHARACTER tip je moguca ovom naredbom. Upotrebom naredbi iz poglavlja 6.14. i 6.15. moguca je konverzija iz bilo kojeg tipa u bilo koji tip podataka.

npr.: Potrebno je izvrsiti konverziju INTEGER tip-a podataka u REAL tip.

```
      INTEGER I/10/
      REAL R
      CHARACTER POM*2
      WRITE(POM,100)I
100   FORMAT(I2)
      READ(POM,200)R
200   FORMAT(F20)
      END
```

6.16. Istrazivanje datoteka

Ako za postojecu datoteku na masovnoj memoriji ne postoje podaci o nacinu otvaranja, oni se mogu istraziti upotrebom naredbe za ispitivanje parametra.

Opci oblik:

```
INQUIRE(UNIT=u,IOSTAT=ios,ERR=s,EXIST=ex,OPEND=od,ACCESS=acc
FORM=fm,RECL=rcl)
```

Koristenje broja datoteke "u" je obavezno. Ostali parametri su proizvoljni.

EXIST=ex "ex" je logicka varijabla i moze primiti vrijednost .TRUE. ako datoteka postoji a vrijednost .FALSE. ako datoteka ne postoji.

OPEND=od je logicka varijabla i moze poprimiti vrijednost

ACCESS=acc je karakter varijabla koja moze poprimiti vrijednost DIRECT ili SEQVENTIAL

FORM=fm je karakter varijabla koja moze poprimiti vrijednost FORMATTED ili UNFORMATTED.

RECL=rcl je cijelobrojna varijabla cija vrijednost je jednaka duzini sloga datoteke "u" ako je ona otvoren kao direktna datoteka.

6.17. Citanje i pisanje matrica

Citanje i pisanje matrica moguce je jednom naredbom READ i WRITE ciji je opci oblik:

opci oblik:

```
READ(UNIT=u,FTM=f,ERR=s,IOSTAT=ios)(M(n),i=n1,n2,n3)...
```

odnosno

```
WRITE(UNIT=u,FTM=f,ERR=s,IOSTAT=ios)(MAT(i,j),j=n1,n2,n3),
i=k1,k2,k3)
```

gdje je:

M(i) i MAT(i,j) jednodimenzionalna odnosno dvodimenzionalna matrica

n1,n2,n3 te k1,k2,k3 su pocetna te krajnja brojna vrijednost i korak kao kod DO-petlje (vidi poglavlj 5.7.).

npr.1: Ucitati iz datoteke 10 podatke u matricu A od 100 clanova ako u jednom slogu ima 10 podataka tipa REAL a jedan podatak se nalazi u polju duljine 8 pozicija sa cetri decimalna mjesta. Ispisati na terminal sve neparne clanove matrice.

rjesenje:

```
      READ (10,1) (A(i),i=1,100,1)
  1   FORMAT (10F8.4)
      WRITE (6,1) (A(i),i=1,100,2)
      END
```

Uocimo da bi isti program napisan upotrebimo DO-petlje izgledao:

```
      DO 5 i=1,100
 5    READ (5,1)A(i)
 1    FORMAT(F8.4)
      DO 10 i=1,100,2
 10   WRITE(6,1)A(i)
      END
```

Pri cemu bi jedan podatak morao biti upisan u jednom slogu, te bi datoteka 10 umjesto 10 slogova imala 100 slogova. Da se ovo izbjegne treba podatke ucitati sa:

```
      DO 10 i=1,10
 10   READ(10,1)A(10*i-9),A(10*i-8),...A(10*i)
 1    FORMAT(10F8.4)
```

sto je prilicno nepregledno.

npr.2: Napisite program koji ce najprije ucitati broj clanova matrice A a potom i samu matricu.

```
 1)     DIMENSION A(100)
        READ(5,100)N, (A(i),I=1,N)
100     FORMAT(I5/(F8.3/))
        END
```

npr.3: Napisite program koji ce istovremeno citati matricu A i B. Obije matrice imaju pet clanova.

```
        DIMENSION A(5),B(5)
        READ (5,100) (A(i),B(i),I=1,5)
100     FORMAT(2F8.3)
        WRITE(6,110) A,B
110     FORMAT(10F8.3)
        END
```

ispis clanova bi bio u sljedecem redosljijedu

A(1),A(2),...,A(5),B(1),...,B(5)

npr.4: Za matricu A(2,2,3) koja bi bila ispisana WRITE naredbom

```
        WRITE(6,110) A
```

redoslijed ispisa bi izgledao:

```
        A(1,1,1),A(2,1,1),A(1,2,1),A(2,2,1)
        A(1,1,2),A(2,1,2),A(1,2,2),A(2,2,2)
        A(1,1,3),A(2,1,3),A(1,2,3),A(2,2,3)
```

prvi indeks se najbrze mijenja.

Zadatak 1.

Kako bi izgledao redoslijed ispisa (ne forma) clanova matrice za:

```
WRITE(6,100)(((A(i,j,k),k=1,3),B(i,j),j=1,2),C(i),i=1,3)
```

Zadatak 2.

Zadana je dvodimenzionalna matrica A(10,15). Od drugog i petog stupca te matrice naciniti matricu B(10,2). Podaci o matrici A ucitavaju se sa terminala po izboru programa.

Zadatak 3.

Ako su za matricu A(25,25) vrijednosti clanaova jednaki produktu indeksa tad vrsiti ispis matrice tako da pise:

```
A(1,1)=1
A(1,2)=2
...
```

ali ispisati samo one clanove cija je vrijednost veca od 100.

Zadatak 4.

Zadana je dvodimenzionalna matrica A(12,12) nadji matricu B(12,2) sastavljenu od dijagonala matrice A.

Za matricu 4X4 raspored indeksa dan je kao na slici 6.1 pa vrijedi da je suma indeksa na sporednij dijagonali i+j=5. Analogno za matricu 12X12 je i+j=13.

```
I-            -I
I 11 12 13 14 I
I 21 22 23 24 I
I 31 32 33 34 I
I 41 42 43 44 I
I-            -I
```

Program :

```
      DIMENSION A(12,12),B(12,2)
C
C  PREPISIVANJE GLAVNE DIJAGONALE (CLAN JE NA GLAVNOJ
C   DIJAGONALI AKO SU INDEKSI JEDNAKI  I=J)
C
      DO 7  I=1,12
      I=J
      B(I,1)=A(I,J)
C
C  PREPISIVANE SPOREDNE DIJAGONALE (CLAN JE NA SPOREDNOJ
C   DIJAGONALI AKO JE   I+J=13
C
      DO 10 I=1,12
      DO 10 J=1,12
      IF(I+J.EQ.13) THEN
         B(I,2)=A(I,J)
         GO TO 10
      ELSE
      END IF
10    CONTINUE
      END
```

7. SPECIFIKACIONE NAREDBE

Specifikacione naredbe opisuju podatke koristene u programu. Specifikacione naredbe jesu: DIMENSION, IMPLICIT, naredbe za eksplicitno deklariranje tipa varijable, EQUIVALENCE, COMMON, PARAMETER, EXTERNAL, INTRINSIC i SAVE. Naredba DIMENSION opisana je u 2.5., naredba IMPLICIT u 2.4.,naredbe za eksplicitno deklariranje tipa varijable u 2.3.. Naredbe EXTERNAL, INTRINSIC, i SAVE opisane su u poglavlju POTPROGRAMI.
Specifikacione naredbe su neizvrsne naredbe. Moraju se nalaziti na pocetku programa prije prve izvrsne naredbe. DATA naredba slijedi nakon svih specifikacionih naredbi.

7.1. EQUIVALENCE naredba

Naredba EQUIVALENCE sluzi u dodijeljivanju dva i vise imena jednom memorijskom registru.

Opci oblik:

EQUIVALENCE(a1,a2,...),(b1,b2,...),...

gdje su:

a1,a2,...,b1,b2,... imena varijabli, matrica, clanova matrica.

U jednoj zagradi se moraju pojaviti najmanje dva imena. Sve varijable u jednoj zagradi zauzimaju iste memorijske registre, te im se dodijeljuje ista vrijednost. Raspored varijabli u zagradi je nebitan. Unutar zagrade mogu biti varijable razlicitih tipova i duzina. Ovo svojstvo moze biti koristeno i za ustedu memorije, jer se u drugom dijelu programa na iste memorijske registre mogu upisati nove vrijednosti preko postojecih koje su bile potrebne u prvom dijelu programa.

Matrice se mogu pojaviti u EQUIVALENCE naredbi. Efekti su isti ako se navedu citave matrice ili samo vodeci clan. Mogu se poistovjetiti i matrice razlicitih dimenzija. Naredba dolazi poslije naredbe DIMENSION a prije izvrsnih naredbi.

Nacin memoriranja podataka uveliko ovisi o racunaru ali programer ne mora poznavati detalje ove organizacije. Za seriju UNIVAC 1100 vrijedi:

Svakom ASCII karakteru je dodijeljena razlicita kombinacija bitova, i naziva se BYTE. Jedan BYTE je jedan karakter. Jedan byte ima 8 bitova. Cetiri byte-a zajedno cine rijec, i to je minimalna kolicina podataka s kojom racunar operira. Za memoriranje vrijednosti varijabli tip-a INTEGER, REAL i LOGICAL koristi se memorijski registar velicine jedne rijeci.
Za varijable tipa DOUBLE PRECISION i COMPLEX koriste se dvije rijeci. Za varijable tipa CHARACTER broj rijeci ovisi o duzini varijable specificirane u naredbi CHARACTER. Ako je duzina varijable CHARACTER*1 zauzeta je prva cetvrtina rijeci, ako je duzina CHARACTER*2 zauzeta je prva polovina rijeci, ako je duzina CHARACTER*4 zauzeta je cijela rijec. Karakter matrica popunjava clanovima memoriju bez obzira na granicu rijeci i to tako da jednu rijec moze okupirati vise clanova u ovisnosti o njihovoj duzini.

Primjer 1: Matrica A od dva clana, sva duzine tri karaktera

```
CHARACTER * 3   A(2)
```

zauzima rijec i pol sto mozemo graficki prikazati kao :

```
-----------------------------------------
I 1. rijec       I 2. rijec       I
-----------------------------------------
I   I   I   I   I   I   I   I   I
-----------------------------------------
I<--A(1)---->I<--A(2)--->I
```

Primjer 2: Naredba EQUIVALENCE moze biti koristena kao:

```
REAL A(5,5),B,C
INTEGER L
EQUIVALENCE (A,B),(C,L)
```

Varijabla B i A(1,1) rasporedene su nad isti memorijski prostor, isto tako i varijable C i L.

Primjer 3:
Dodjeljivanje iste memorije CHARACTER varijablama

```
CHARACTER A*4,B*4,C(2)*3
EQUIVALENCE (A,C(1)),(B,C(2))
```

```
-----------------------------------------
I   1 RIJEC      I   2 RIJEC      I
-----------------------------------------
I   I   I   I   I   I   I   I   I
-----------------------------------------
I<---C(1)-->I<---C(2)-->I          zauzima matrica C
I<-------A------->I                zauzima varijabla A
            I<-------B------->I    zauzima varijabla B
```

U ovom slucaju cetvrti karakter od varijable A jednak je prvom karakter varijable B.

Primjer 4:

```
INTEGER M(20)
REAL AA(10,9)
EQUIVALENCE (AA(1,9),M(1))
```

Matrica AA i M imaju 10 clanova koji zauzimaju istu memorijsku poziciju. To su clanovi devetog stupca matrice AA i prvih deset clanova niza M.

Primjer 5:
Ako se u pocetku programa koristi matrica C(10,10), a kasnije ona nije potrebna, vec matrice B(10) i A(5,5). Matrice A i B ne smije se staviti nad istu memorijsku zonu.

Matricu C moze se prekriti na slijedeci nacin:

```
DIMENSION A(5,5),B(10),C(10,10)
EQUIVALENCE ((C(1,1),A(1,1),(C(3,6),B(1))
```

Primjer 5:
Ako je dano u programu:
DIMENSION A(10),B(5),C(2,2)
EQUIVQLENCE (A(5),B(1),C(1,1))
tada dana naredba EQUIVALENCE ima isti efekt kao i naredba:
EQUIVALENCE (A(7),B(3),C(1,2))
jer se u oba slucaja prekrivaju clanovi:

```
A(1) A(2) A(3) A(4) A(5)   A(6)   A(7)   A(8)   A(9) A(10)
                    B(1)   B(2)   B(3)   B(4)   B(5)
                    C(1,1) C(2,1) C(1,2) C(2,2)
```

Primjer 7:

```
REAL A(10,10),B,C,D(10)
INTEGER N(20)
EQUIVALENCE (A(3,3,),B,D(1)),(B,C,N(1))
```

Posto se varijabla B pojavljuje u obe zagrade, tad sve varijable dijele istu memorijsku poziciju

7.2. COMMON naredba

Omogucuje da u vise programskih jedinica prenosimo grupu varijabli.

opci oblik :

```
COMMON / c1 / n1,n2,... / c2 / m1,m2,... ...
```

gdje je :

c1,c2 - naziv zajednicke ZONE podataka izmedju vise programa. ZONA c1 mora biti imenovana po pravilima za imenovanje varijabli i pise se izmedju kosih crta.

n1,n2 - imena varijabli ili matrica koje su smjestene u zajednicku zonu podataka.

Primjer 1:

```
COMMON /ABC/A,X,C,I11/ COM1/C(5),KA,A(10)
```

Varijable A,X,C i I11 su smjestene u dio memorije koji nosi ime ABC a matrice C i A i varijabla KA su smjestene u dio memorije koji smo proglasili zajednickim imenom COM1.

Ako zonu ne zelimo imenovati, sto je dozvoljeno, tada uzastopno pisemo dvije kose crte, a ako je prvi blok neimenovan i njih mozemo ispustiti.

Primjer 2:

```
COMMON  C11,A3/Z1/A(10),Y(5)//C12,A7(3)/Z2/Y,AZ
```

Postoje dvije imenovane zone Z1 i Z2 sa pripadnim varijablama i jedna neimenovana zona sa varijablama C11,A3,C12 i nizom A7(3)

Dozvoljena je upotreba vise COMMON naredbi sa razlicitim listama. Naredba COMMON je opisna naredba u programu.

U svim programskim jedinicama ime zajednicke zone koja se zeli prenijeti ostaje nepromijenjeno a imena varijabli se mogu proizvoljno mijenjati. Redosljed varijabli je bitan, jer u istoj zajednickoj zoni redom varijablama odgovaraju iste vrijednosti.

Varijabla koja se koristi u jednoj programskoj jedinici (programska jedinica je opisana u poglavlju POTPROGRAMI) naziva se lokalna varijabla, i njoj se moze pristupiti samo iz te programske jedinice. COMMON naredba omogucuje imenovanje globalnih varijabli, kojima se moze pristupati iz vise programskih jedinica. Postojanje globalnih varijabli u programu moze stvoriti znacajne probleme jer onemogucuje otkrivanje mjesta nastanka greske. Uzmimo da niz programskih jedinica naizmjenicno pristupa globalnoj varijabli i neka u jednoj programskoj jedinici dodje do greske te i vrijednost globalne varijable bude pogresna a ta se greska ustanovi ne u toj vec u nekoj drugoj programskoj jedinici. Trazenje greske u programskoj jedinici gdje je ona uocena ne daje rezultate vec je potrebno analizirati sve programske jedinice koje su do tad mogle pristupati globalnoj varijabli.

Povezanost programskih jedinica podacima preko globalnih varijabli je i velika poteskoca pri odrzavanju i modifikaciji programa. Opca preporuka je:

NE KORISTITI NAREDBU COMMON

Prenos vrijednosti varijabli izmedju programskih jedinica vrsite pomocu argumenata kod poziva programskih jedinica (opisano u poglavlju POTPROGRAMI).

7.3. PARAMETAR naredba

Omogucuje proglasavanje simbolickog imena konstantom.

Opci oblik:

PARAMETAR (n1=e1 ,n2=e2,...)

gdje je:

n1,n2,... simbolicka imena konstanti

e1,e2,... vrijednosti konstante (izraz sacinjen od konstanti)

Ovo je neizvrsna naredba i u FORTRAN-u 77 mora se pojaviti prije prve :DATA naredbe, funkcijske naredbe i izvrsne naredbe. Simbolicko ime moze biti tipa INTEGER, REAL, DOUBLE PRECISION, COMPLEX i tad dodjeljena vrijednost moze biti aritmeticki izraz. Ako je simbolicko ime tipa CHARACTER ili LOGICAL tad i odgovarajuci izrazi mora biti tipa CHARACTER ili LOGICAL. U PARAMETAR naredbi se mogu koristiti simbolicka imena konstanti definirana u prethodnim PARAMETAR naredbama. U istojm programskoj jedinici vrijednost parametar konstante se ne smije mijenjati. Nije dozvoljena upotreba standardnih funkcija (vidi tabelu u Dodatak C.).Potenciranje je dozvoljeno samo ako je eksponent tipa INTEGER . Parametar konstanta se ne smije koristiti u FORMAT naredbi i u specifikacionim naredbama za odredjivanje duzine varijabli izuzev u CHARACTER naredbi ali pisanjem parametar konstante u zagradama.

Primjer 1: Koristenja PARAMETAR naredbe.

```
INTEGER A,B,C
PARAMETAR (A=3)
PARAMETAR (IB=2**18)
PARAMETAR (B=4*A , C=A+B )
D = IB + B*ALOG(C)
  .
  .
  .
```

Varijable A, IB, B i C su poprimile vrijednosti 3, 36, 12 i 15 za vrijeme prevodjenja programa iz simbolickog u masinski kod a naredba D = IB + B*ALOG(C) u programu ima isti efekat kao da pise : D = 36+12*ALOG(15).

8. POTPROGRAMI

U ovom poglavlju je opisana organizacija FORTRAN-skih programa i dana su pravila za njihovo pisanje.

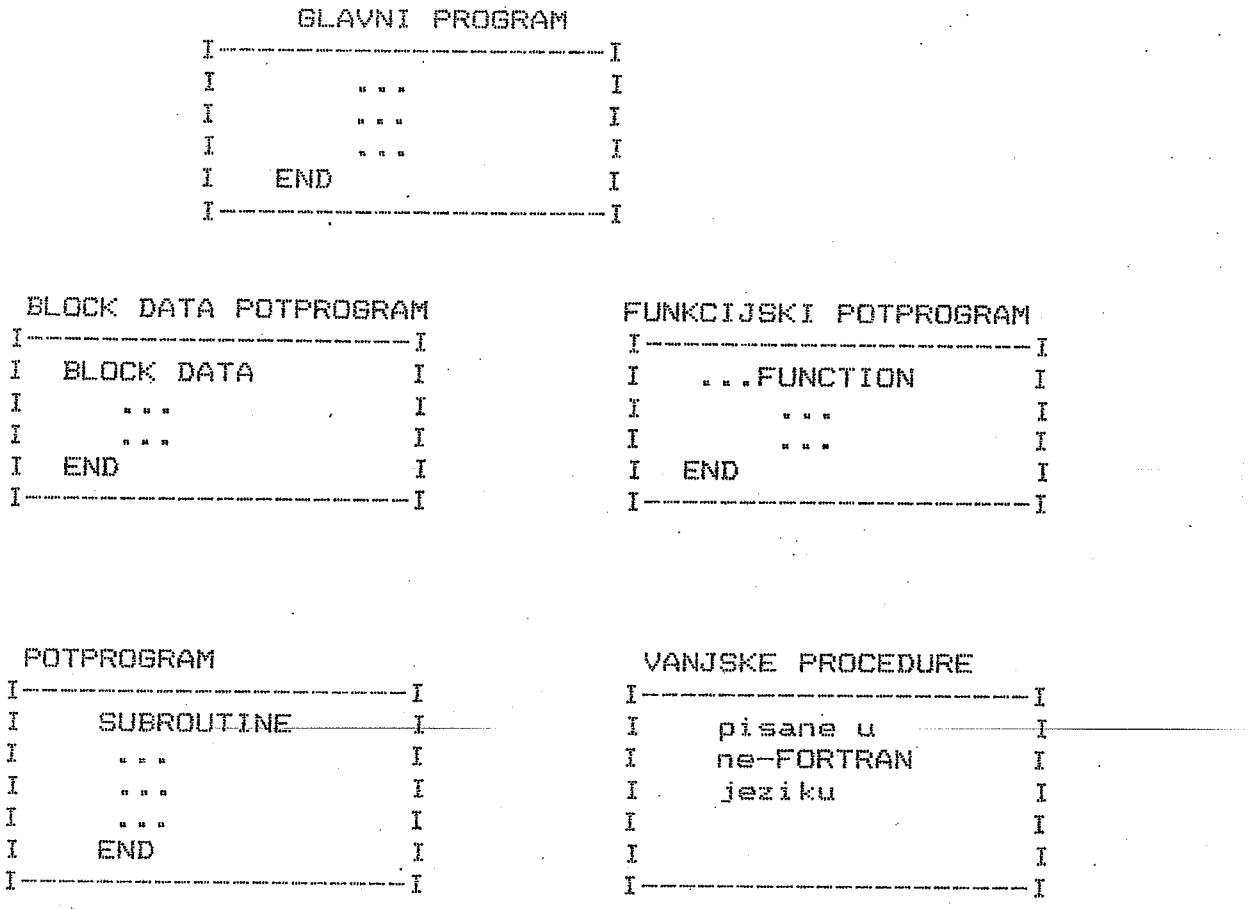
FORTRAN program je sacinjen od jednog i samo jednog glavnog programa i vise potprograma i vanjskih procedura. Glavni program definira jasno osnovne korake za rjesavanje datog problema. I glavni program i potprogram nazivamo programskom jedinicom. Potprogram je podredjena programska jedinica glavnom programu. Ukoliko u jednom programu imamo potprograme tada se glavni program mora nalaziti fizicki ispred potprograma.

Svaka programska jedinica se sastoji od naredbi i od komentara. Jedna naredba moze biti napisana u jednoj ili vise linija. Linije komentara nisu izvrsne i sluze za pojasnjavanje programa.

Tipovi programskih jedinica jesu : glavni program, funkcijski potprogram, potprogram, vanjske procedure i BLOCK DATA potprogram. (vidi sliku 8.1)

Glavni program, funkcijski potprogram i potprogram cini niz naredbi i komentara. BLOCK DATA programska jedinica je niz specifikacionih naredbi i komentara. Vanjske procedure su dijelovi programa pisani u nekom drugom programskom jeziku.

Razlicite programske jedinice mogu biti upisane u jednu ili vise datoteka masovne memorije racunara. FORTRAN prevodilac i kolektor prevode izvorni kod (SOURCE PROGRAM) u masinski kod (OBJECT PROGRAM) i smjestaju ga u jednu datoteku. Pozivom imena datoteke u kojoj je smjesten OBJECT PROGRAM zapocinje izvrsavanje programa na racunaru.



sl. 8.1.

Potprogrami najcesce predstavljaju niz naredbi koje bi se pojavljivale na vise mjesta jednog programa za isto izracunavanje ali za razlicite vrijednosti argumenata. U FORTRAN prevodioc su ugradeni interni potprogrami koje nazivamo STANDARDNE FUNKCIJE (vidi Dodatak C.) a koji stoje programu na raspolaganju.

Upotreba potprograma pri programiranju ima prednosti kao sto su: kraci zapis programa, manja mogucnost greske, lakse testiranje programa, isti potprogram se moze koristiti u raznim programima, modularna izgradnja software.

8.1. PROGRAM naredba

Sluzi za imenovanje glavnog programa

opci oblik:

PROGRAM ime glavnog programa

Ova naredba je opciona,te se moze i ne mora koristiti. Ako se koristi onda se mora pojaviti kao prva linija u glavnom programu. Sluzi za dokumentaciju programa.

8.2. Naredbe za definiranje funkcije

Sluzi za definiranje imena funkcije s odredenim izrazom.

opci oblik:

N(a,b,c,...)=izraz

gdje je:

N — ime definirane funkcije. Vrijede pravila za imenovanje varijabli.

a,b,c — privremeni argumenti koristeni u izrazu. Maximalno je dozvoljeno 150 argumenata. Kod imenovanja vrijede pravila za imenovanje varijabli.

izraz — FORTRAN izraz koji definira nacin izracunavanja vrijednosti funkcije preko privremenih argumenata.

Sve naredbe za definiranje funkcije moraju se pojaviti prije prve izvrsne naredbe u programskoj jedinici, a poslije svih drugih specifikacionih naredbi. Svi navedeni argumenti moraju se pojaviti u navedenom izrazu. Izraz moze sadrzavati druge prije definirane funkcije. Konstante i stvarne vrijednosti se mogu pojavit u izrazu.

Na mjestu potrebnom za izracunavanje vrijednosti ranije definirane funkcije, dovoljno je pisati ime funkcije sa stvarnim argumentima kao

opci oblik:

N(x,y,z,...)

gdje je:

N — ime definirane fukcije. Vrijede pravila za imenovanje varijabli.

x,y,z,... — stvarni argumenti s kojima zelimo da se izracuna vrijednost ranije definirane funkcije. To mogu biti izrazi ili imena matrica.

Poredak, broj i tip stvarnih argumenata u pozivu funkcije mora biti isti kao kod definiranja funkcije.

Ako su "izraz" i ime funkcije "N" oba aritmetickog tipa, rezultat izraza ce biti preveden u tip od "N" u skladu s pravilima za rjesavanje aritmetickog izraza (vidi 6.1. ARITMETICKE NAREDBE).

Ako su "izraz" i "N" tipa CHARACTER duzina "izraza" ce biti promijenjena, ako je razlicita, u duzinu od "N".

Primjer 1: Napisati program za izracunavanje aritmeticke sredine tri realna broja koristenjem naredbi za definiranje funkcija.

```
      REAL I,J,B,ARITS,AS
      ARITS(X,Y,Z)=(X+Y+Z)/3
10    READ(5,100,ERR=200)I,J,B
100   FORMAT(3F5.1)
      AS=ARITS(I,J,B)
      PRINT *,' I=',I,' J=',J,' B=',B,' AS=',AS
      GO TO 10
200   END
```

Primjer 2:

```
C  DEFINICIJA FUNKCIJE
      N(I,J)=K+L*I+M*J
      DATA K,L,M/1,2,3/
C  POZIV FUNKCIJE (ARGUMENTI SU KONSTANTE)
      IZLAZ1=N(4,5)+3
      I1=5
      J1=4
C  POZIV FUNKCIJE ARGUMENTI SU VARIJABLE
      IZLAZ2=N(J1,I1)+3*(N(12,M)-2)
      END
```

8.3. Poziv potrprograma

Naredba CALL omogucuje poziv na izvrsenje potprograma iz neke programske jedinice.

opci oblik:

```
CALL ime(a,b,c,...)
```

gdje je:

ime — naziv potprograma kojeg pozivamo na izvrsenje.

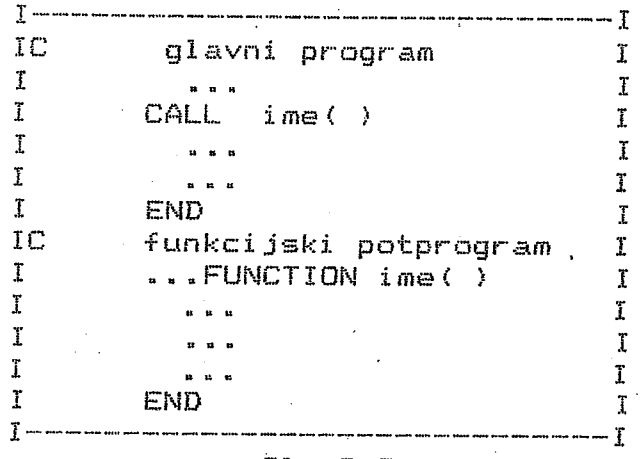
a,b,c,... — su argumenti koji se prenose u potprogram. Argumenti mogu biti FORTRAN izrazi, matrice, funkcije ili labele u programu. Broj argumenata ne smije biti veci od 150.

Ako su argumenti labele onda je ispred broja labele "n" obavezno pisati "*", odnosno opci oblik argumenta ima oblik "*n" gdje je "n" labela u istoj programskoj jedinici.

Kontrola redoslijeda izvrsenja naredbi nakon izvrsenja potprograma, prenosi se na prvu izvrsnu naredbu iza CALL naredbe. Ako su argumenti labele onda se kontrola moze prenijeti na bilo koji dio programa, odnosno na naredbu s odgovarajucom labelom (vidi SUBROUTINE program).

8.4. Funkcijski potprogram

To je zasebna programska jedinica koja zapocinje naredbom FUNCTION a moze zavrsiti sa END naredbom. Polozaj funkcijskog potprograma unutar glavnog programa je prikazan slikom:



Sl. 8.2.

Funkcijski potprogram ne smije sadrzavati naredbe: SUBROUTINE, PROGRAM i BLOCK DATA.

Funkcijski potprogram ne smije sadrzavati poziv samog sebe na izvrsenje bilo direktno, bilo iz nekog njemu podredenog funkcijskog potprograma.

Naredba za obavjestavanje prevodioca da grupa naredbi koja slijedi pripada funkcijskom potprogramu ima

opci oblik:

type FUNCTION ime (a,b,c,...)

gdje je:

type — sluzbena rijec iz grupe: INTEGER, REAL, DOUBLE PRECISION, COMPLEX, CHARACTER ili LOGICAL. Moze se i ne mora koristiti za eksplicitno definiranje tipa funkcije.

ime — je naziv funkcijskog potprograma

a,b,c,... — su imena argumenata koji mogu biti varijable ili matrice. Najvise je dozvoljeno 150 argumenata.

Naredba FUNCTION mora biti prva u funkcijskom potprogramu odredjuje ime, argumente i, ako je potrebno, tip funkcije. Pri pozivu funkcijskog potprograma na izvrsenje, iz razlicitih programskih jedinica, tip funkcije mora biti isti kao i tip u funkcijskom potprogramu. Nakon zavrsetka funkcijskog potprograma se odredjena vrijednost dodijeljuje se odredjena vrijednost funkciji 'ime ' i ta se vrijednost prenosi u programsku jedinicu koja je zahtijevala izvrsenje funkcijskog potprograma.

(Vidi primjer kod 8.5. RETURN naredba).

8.5. RETURN naredba

Sluzi za povratak kontrole izvrsenja naredbi iz potprograma u program. Kontrola se prenosi na prvu slijedecu izvrsnu naredbu iza naredbe za poziv potprograma na izvrsenje.

opci oblik:

```
RETURN n
```

gdje je:

n broj ili cjelobrojni izraz.

Za funkcijske potprograme ne koristi se "n". Pri izvrsenju naredbe RETURN kontrola se prenosi u programsku jedinicu odakle je funkcijski potprogram pozvan na izvrsenje.

Za SUBROUTINE potprogram moze se i ne mora koristiti izraz n. Ako se upotrebljava, onda se kontrola prenosi na odgovarajucu labelu navedenu u pozivu SUBROUTINE programa. Ako su ukupno navedene naprimjer tri labele (vidi poglavlje Poziv potprograma) onda mora biti n <= 3.

Primjer funkcijskog potprograma:

```
C   tip funkcije AFS nije eksplicitno deklariran pa se
C   podrazumijeva REAL
C
      FUNCTION AFS(I,A,B,C)
C   ako je I=0 izracunaj vrijednost funkcije AFS=a/b
C   ako je I=1 idi na labelu 3
C   ako je I=2 idi na labelu 4
      GO TO (3,4)I
      AFS = A/B
      RETURN
      END
3     AFS=ABS(C-A/B)
      RETURN
4     C=0
      AFS=A+B+C
      RETURN
      END
C   vrijednosti argumenata prenesene iz glavnog programa sluze za
C   racunanje vrijednosti funkcije AFS i nakon racunanja u glavni
C   program se vraca vrijednost AFS
```

Glavni program koji bi pozvao ovaj funkcijski potprogram na izvrsenje mogao bi imati oblik:

```
      DATA I,A,B,C/3.,4.,5.,6./
C   prvi poziv funkcijskog potprograma
      CALL AFS(I,5,B,C)
C   drugi argument je prenesen kao konstanta
      D=AFS-1.
C   s funkcijom se operira kao s varijablom
C
C   drugi poziv funkcije - svi argumenti su konstante
      CALL AFS(1,2,3,4)
      U=MAX(D,AFS)
      END
```

8.6. SUBROUTINE potprogram

Kao i funkcijski potprogram SUBROUTINE potprogram je odvojena programska jedinica, koja zapocinje naredbom SUBROUTINE a zavrsava END naredbom. SUBROUTINE potprogram moze imati druge funkcijske ili druge SUBROUTINE potprograme. Potprogram ne moze pozvati samoga sebe bilo direktno bilo iz nekog svog potprograma. Naredba SUBROUTINE kojom mora poceti potprogram ima

opci oblik:

SUBROUTINE ime(a,b,c,...)

gdje je:

ime je ime potprograma po kome se prepoznaje u drugoj programskoj jedinici. Imenu se ne dodijeljuje neka vrijednost kao kod funkcijskog potprograma i prvo slovo nema neko znacenje.

a,b,c,... su formalni argumenti koji mogu biti varijable,matrice ili znak " * ". Jedan argument se moze pojaviti u listi argumenata samo jednom. Broj argumenata ne smije biti veci od 150. Ako potprogram nema potrebe za prenosom argumenata zagrade se mogu izostaviti.

Naredba SUBROUTINE mora biti prva u potprogramu. Ona odreduje ime i argumente potprograma. Iz glavnog programa u potprogram moze biti prenijeti vise varijabli i obrnuto. Argumenti mogu s odredenom vrijednoscu doci u potprogram i ne, odnosno mogu dobiti vrijednost tek u potprogramu. U glavni program se vraca vrijednost svih argumenata. Na ovaj nacin programer sam odreduje koliko i koje argumente ce prihvatiti kao rezultat rada potprograma. Prikazimo na jednostavnom primjeru upotrebu potprograma.

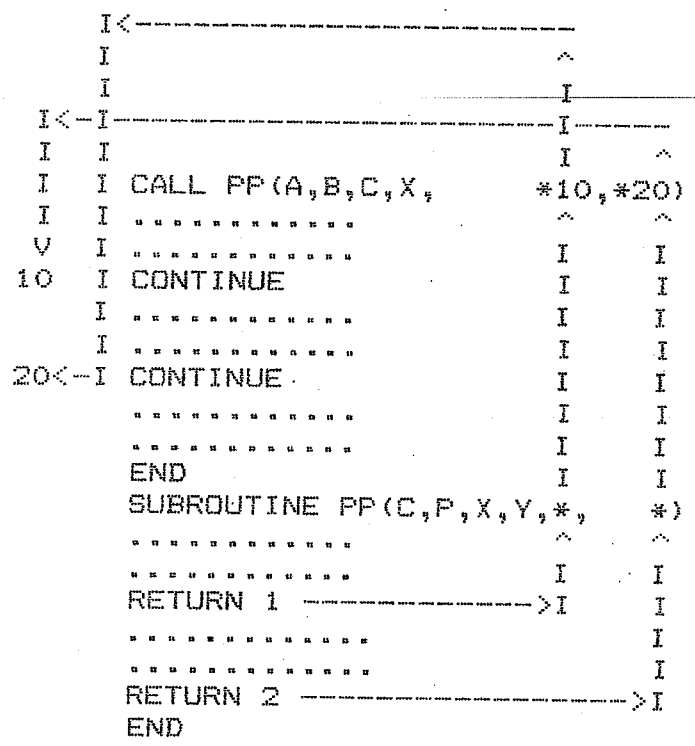
npr.1:

Napisi opci potprogram za racunanje aritmeticke sredine niza X od 100 clanova.

```
C  glavni program
       DIMENSION X(100)
       READ(5,100)N,X
100    FORMAT(I3/(5F8.2))
C  poziv potprograma, X i N ulaze u potprogram dok je AS rezultat
C  rada potprograma
       CALL ARSR(X,N,AS)
       PRINT *,'AS=',AS
       END
C
C
C  potprogram
C
       SUBROUTINE ARSR(A,I,C)
       DIMENSION A(1)
       SUM=0
       DO 5 J=1,I
5      SUM=SUM+A(J)
       AS=SUM/I
       RETURN
       END
```

Ako se medju argumentima u pozivu potprograma pojavi zvijezdica "*" onda se kontrola u glavni program prenosi na naredbu s labelom "n" (vidi CALL i RETURN naredbu). Medju argumentima kod naredbe SUBROUTINE moze biti vise zvjezdica. U potprogramu se tad javljaju naredbe RETURN 1, RETURN 2, itd. do RETURN n gdje "n" predstavlja n-tu labelu u pozivu potprograma. Odgovarajuca CALL naredba ima n argumenata koji su labele. Ako se potprogram zavrsi na naredbi RETURN , kontrola se prenosi u glavni program na prvu izvrsnu naredbu iza CALL naredbe. Ako se potprogram zavrsi na naredbi RETURN 1, pronalazi se medju listom argumenata prva zvjezdica u SUBROUTINE naredbi pa zatim u CALL naredbi argument na istoj poziciji kao i zvjezdica i kontrola se prenosi na naredbu u glavnom programu s labelom odgovarajuceg argumenta. RETURN 2 odgovara drugoj zvjezdici odnosno drugoj labeli itd.

Prikazimo ovo graficki:



Strelice prikazuju smjer prenosa kontrole iz potprograma u glavni program.

Privremeni argumenti preneseni kao argumenti naredbom SUBROUTINE ne smiju se pojaviti u naredbama: EQUIVALENCE, COMMON, DATA, PARAMETER, SAVE i INTRISTIC.

8.7. ENTRY naredba

Naredba ENTRY definira mjesto pocetka izvrsenja SUBROUTINE potprograma ili FUNCTION potprograma.

opci oblik:

```
ENTRY ime(a,b,c,...)
```

gdje je:

ime je ime potprograma (vidi naredbu SUBROUTINE ili FUNCTION)

a,b,c su formalni argumenti (vidi naredbu SUBROUTINE ili FUNCTION)

Nikoja dva argumenta u listi argumenata ne smiju biti istog imena.

Programska jedinica moze sadrzavati vise ENTRY naredbi. Na taj nacin definira se vise mogucih mjesta ulaska u programsku jedinicu. Razna mjesta mogu biti pozivana naredbom CALL. Najvise je dozvoljeno 511 ENTRY naredbi u jednoj programskoj jedinici. Potprogram ne smije pozivati sam sebe, ali moze pozivati ENTRY naredbe navedene u njemu. ENTRY ne moze pozivati sam sebe. Ako se pri izvrsenju programa naidje na ENTRY naredbu (moze ih biti vise u programskoj jedinici), ona se preskace i izvrsava se slijedeca izvrsna naredba u sekvenci. ENTRY naredba se ne smije pojaviti unutar DO -petlje ili IF-BLOK strukture.

Primjer: Neka se potprogram BROD sastoji od niza tocaka ulaza koje mogu biti pozvane od strane programa.

```
      COMMON A1/X,Y,Z/
        ...
        ...
C poziv potprograma BROD od mjesta LIM
      CALL LIM
        ...
        ...
C poziv potprograma BROD od mjesta KRMA
      CALL KRMARA(U1,U2,U3)
        ...
        ...
C poziv potprograma BROD od mjesta LIM
      CALL LIM
        ...
        ...
C poziv potprograma BROD od mjesta TRUP
      CALL TRUP(U,V)
      END
      SUBROUTINE BROD
      COMMON A1/A,B,C/
      ENTRY TRUP(U,V)
        ...
        ...
      ENTRY LIM
        ...
        ...
      RETURN
      ENTRY KRMARA(U1,U2,U3)
        ...
        ...
      RETURN
      END
```

8.8. BLOCK DATA potprogram

BLOCK DATA potprogram sadrzi podatke koji stoje na raspolaganju programu.

opci oblik:

BLOCK DATA ime

gdje je:

ime je opci naziv grupe podataka iza naredbe BLOCK DATA do END naredbe. Ime mora biti jedinstveno. U jednom programu moze biti samo jedan neimenovani BLOCK DATA.

BLOCK DATA se sastoji od niza specifikacionih naredbi i komentara.

Primjer: Neka je u BLOCK DATA s imenom MARKO inicijalizirana vrijednost matrice N od deset clanova i u COMMON bloku s imenom UZMI se omogucuje prijenos na zeljeno mjesto u programu.

```
BLOCK DATA MARKO
DIMENSION N(10),A(3)
COMMON /UZMI/N,AB
DATA N/3,4,2,1,6,5,8,7,0,9/
END
```

U bloku UZMI prenosi se i matrica AB a njena vrijednost nije inicijalizirana.

8.9. SAVE naredba

Sluzi za memoriranje vrijednosti u potprogramu i nakon izvrsenja RETURN ili END naredbe. Ponovnim ulaskom u potprogram, sve varijable definirane sa SAVE sadrze vrijednosti koje su imale kod zadnjeg izvrsenja potprograma.

opci oblik:

SAVE a,b,c,...

gdje su:

a,b,c,... su imena varijabli,matrica ili COMMON bloka (izmedju dvije kose crte).

SAVE naredba je neizvrsna naredba.

U drugim programskim jedinicama se ne mogu mijenjati vrijednosti varijabli definiranih u SAVE naredbi izuzev onih iz COMMON bloka.

Naredba nema efekta u glavnom programu.

ZADACI

1. Napisati program za rjesavanje kvadratne jednadzbe
a*x**2 + b*x + c = 0

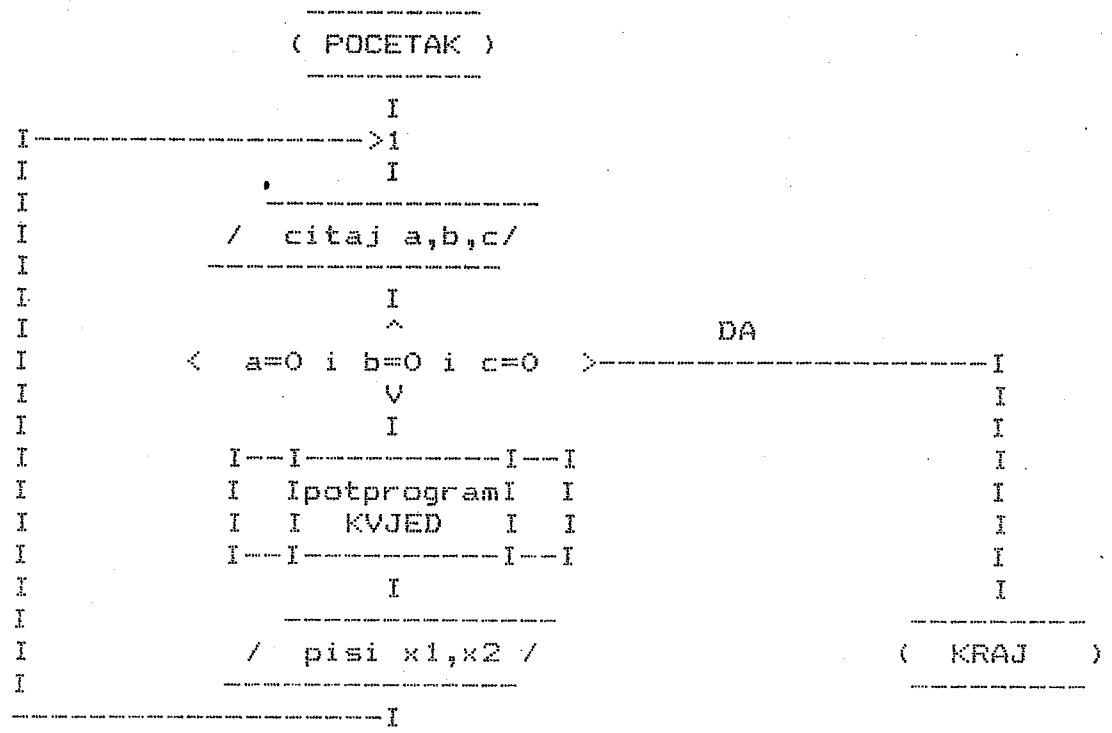
Koeficijenti a,b,c se ucitavaju sa tastature i mogu biti realni brojevi s tri decimalna i pet cijelih mjesta.
Rjesenje kvadratne jednadzbe je dano izrazom:

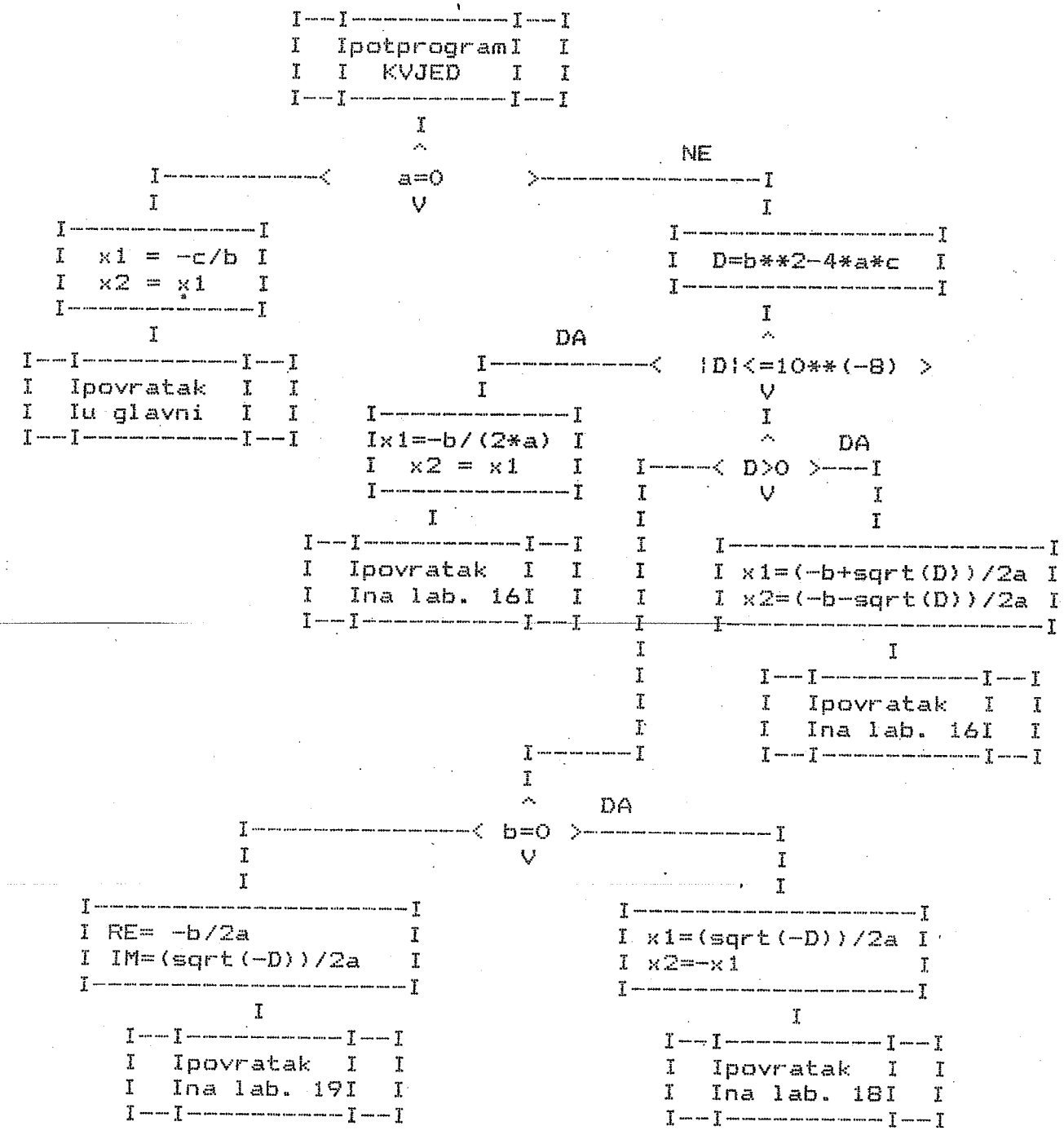
x1,2 = ((-b +- (b**2 - 4*a*c)**(1/2))/ 2*a

U glavnom programu ucitati koeficijente i ispisati rezultat. U potprogramu " KVJED " naci rjesenje jednadzbe.

U slucaju da je :

a) a=0 rjesenje je: x1=x2=-c/b
b) determinanta D=(b**2-4*a*c)**(1/2)=0 ==========>
rjesenja su x1=x2 = -b/(2*a)
c) D>0 ==========> postoje dva realna i razlicita rjesenja
d) D<0 ==========> rjesenja su kompleksna i to:
ako je b=0 rjesenja su imaginarna x1=+i*IM
x2=-i*IM
ako je b razlicito od 0 rjesenja su konjugirano kompleksna
x1= RE + i*IM
x2= RE - i*IM
pri cemu je: RE=-b/2*a i IM=(-D)**(-1/2)/(2*a)
Vrijednost determinante D smatramo da je jednaka nuli ako je ABS(D) < = 10E-8. Ako su sva tri koeficijenta a,b,c jednaka nuli zavrsi program, a nakon racunanja i ispisa rezultata usmjeriti program na ucitavanje novih koeficijenata.





Sl. 8.3.

PROGRAM KVADR.JEDNADZBA

```
C        GLAVNI PROGRAM
         REAL X1,X2,RE,IM,A,B,C
C          UCITAVANJE KOEFICIJENATA
1        WRITE(6,101)
101      FORMAT(' UPISITE KOEFICIJENT a')
         READ(5,100,ERR=99)A
100      FORMAT(F9.3)
         WRITE(6,102)
102      FORMAT(' UPISITE KOEFICIJENAT b')
         READ(5,100,ERR=99)B
         WRITE(6,103)
103      FORMAT(' UPISITE KOEFICIJENT c')
         READ(5,100,err=99)C
C ISPITIVANJE PREKIDA RADA
         if((a.eq.0).and.(b.eq.0).and.(c.eq.0))then
         stop  'koeficijent a=b=c=0  prekidam rad'
```

```
      end if
C
C POZIV POTPROGRAMA
C
      call kvjed(a,b,c,x1,x2,re,im,*16,*17,*18,*19)
C
      write(6,201)a,b,c,x1,x2
201   format(/5x,'rjesenja jednaka za a =',f9.3,2x,b=',f9.3,2x,
     +'c=',f9.3,2x,'su  X1=',f9.3,2x,'X2=',f9.3)
      go to 1
16    write(6,201)a,b,c,x1,x2
      go to 1
17    write(6,202)a,b,c,x1,x2
      go to 1
202   format(/5x,'rjesenja realna i razlicita za a=',f9.3,2x,'b='
     +,f9.3,2x,'c=',f9.3,2x,'su  x1=',f9.3,2x,'x2=',f9.3)
18    write(6,203)a,b,c,x1,x2
203   format(/5x,'rjesenja imaginarna za a=',f9.3,2x,'b=',f9.3,2x
     +,'c=',f9.3,2x,'su   x1=',f9.3,2x,'x2=',f9.3)
      go to 1
19    write(6,204)a,b,c,re,im,re,im
204   format(/5x,'rjesenja konjugirano kompleksna za a=',f9.3,2x,
     +'b=',f9.3,2x,'c=',f9.3,2x,'su  x1=',f9.3,2x,'x2=',f9.3)
      go to 1
```

```
99    write(*,205)a,b,c
205   format(//5x,'***greska***kod upisa koeficijenata a=',
     +'upisite ponovo')
      go to 1
      end
C
C
C
C       potprogram KVJED
      subroutine kvjed(a,b,c,x1,x2,re,im,*,*,*,*)
C        da li je linearna jednadzba?
      if(a.eq.0)then
        x1=-c/b
        x2=x1
        return
      end if
C       racunanje determinante
      d=b**2-4*a*c
C       da li je determinanta jednaka nuli  (manja od 10**(-8))
      if(abs(d).le.1.e-8)then
        x1=-b/(2*a)
        x2=x1
        return 1
      end if
c       da li je determinanta veca od nule?
      if(d.gt.0)then
        x1=(-b+sqrt(d))/(2*a)
        x2=(-b-sqrt(d))/(2*a)
        return 2
      end if
c       preostaje jos da su rjesenja kompleksns
c       da li je b=0? znaci imaginarna rjesenja
      if(b.eq.0)then
        x1=sqrt(-d)/(2*a)
        x2=-x1
        return 3
      else
c       preostaje samo konjugirano kompleksno rjesenje
        re=-b/(2*a)
        im=sqrt(-d)/(2*a)
        return 4
```

```
end if
end
```

Da se izbjegne inverzija u upravljanu, gdje potprogram odredjuje koji dio glavnog programa ce se izvrsavati, predlaze se ispis rezultata na onom mjestu gdje se rezultati izracunaju, a ne njihov ispis u glavnom programu. Kako bi glavni program bio jos citljiviji predlaze se ucitavanje koeficijenata realizirati u zasebnom potprogramu. U tom slucaju glavni program ima samo upravljacku funkciju te je i najuocljivije sta program u cijelini radi.

SIMBOL	OKTALNA VRIJEDNOST	SIMBOL	OKTALNA VRIJEDNOST
PRAZNINA (b)	40	P	120
!	41	Q	121
"	42	R	122
#	43	S	123
$	44	T	124
%	45	U	125
&	46	V	126
'	47	W	127
(	50	X	130
)	51	Y	131
+	52	Z	132
*	53	[	133
,	54	\	134
-	55	]	135
.	56	^	136
/	57	_	137
0	60	`	140
1	61	a	141
2	62	b	142
3	63	c	143
4	64	d	144
5	65	e	145
6	66	f	146
7	67	g	147
8	70	h	150
9	71	i	151
:	72	j	152
;	73	k	153
<	74	l	154
=	75	m	155
>	76	n	156
?	77	o	157
@	100	p	160
A	101	q	161
B	102	r	162
C	103	s	163
D	104	t	164
E	105	u	165
F	106	v	166
G	107	w	167
H	110	x	170
I	111	y	171
J	112	z	172
K	113	{	173
L	114	!	174
M	115	}	175
N	116	~	176
O	117	DELETE	177

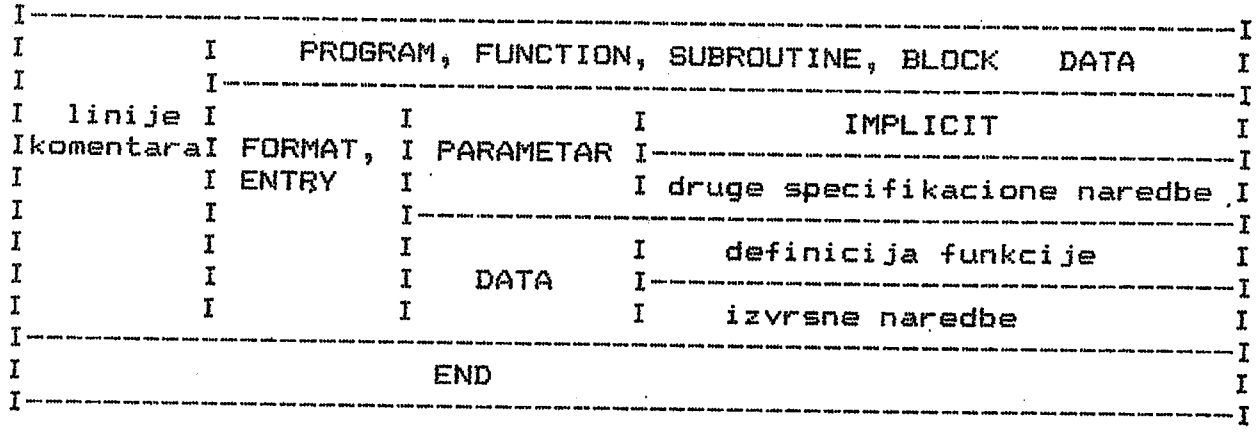
vrsta naredbe	naziv naredbe
Specifikacione naredbe	DIMENSION COMMON EQUIVALENCE EXTERNAL PARAMETER INTRINSIC IMPLICIT INTEGER, REAL, DOUBLE PRECISION, COMPLEX, LOGICAL, CHARACTER SAVE
Za inicijalizaciju	DATA
Format naredba	FORMAT
Naredbe za definiranje funkcije	
Imenovanje programskih cijelina	PROGRAM FUNCTION SUBROUTINE ENTRY BLOCK DATA
Za dodijeljivanje	Aritmeticke, logicke i karakteral naredbe dodijeljivanja ASSIGN
Kontrolne naredbe	GO TO IF (IF,THEN,ELSE,ELSE IF,END IF) CALL CONTINUE RETURN STOP PAUSE DO END
Ulazno/izlazne	READ WRITE PRINT REWIND BACKSPASE ENDFILE OPEN CLOSE INQUIRE

OPIS FUNKCIJE	MATEMATICKI OBLIK	IME FUNKC.	ARGUMENT (BROJ i TIP)	FUNKCIJA TIP
Prirodni logaritam	ln(x) (x>0)	ALOG	1 REAL	REAL
		DLOG	DOUBLE	DOUBLE
		CLOG	COMPLEX	COMPLEX
Dekadski logaritam	log (x) (x>0)	ALOG10	REAL	REAL
	10	DLOG10	DOUBLE	DOUBLE
Eksponencijalna funkcija	e^x	EXP	1 REAL	REAL
		DEXP	DOUBLE	DOUBLE
		CEXP	COMPLEX	COMPLEX
Kvadratni korijen	$\sqrt{x}$ (x>=0)	SQRT	1 REAL	REAL
		DSQRT	DOUBLE	DOUBLE
		CSQRT	COMPLEX	COMPLEX
Arkussinus dan u radijanima	Arcsin(x) -1<=x<=1	ASIN	1 REAL	REAL
		DASIN	DOUBLE	DOUBLE
Arkuscosinus dan u radijanima	Arcos(x) -1<=x<=1	ACOS	1 REAL	REAL
		DACOS	DOUBLE	DOUBLE
Arkustangens dan u radijanima	Arctan(x) -1<=x<=1	ATAN	1 REAL	REAL
		DTAN	DOUBLE	DOUBLE
Sinus	sin(x) x u radijanima	SIN	1 REAL	REAL
		DSIN	DOUBLE	DOUBLE
		CSIN	COMPLEX	COMPLEX
Cosinus	cos(x) x u radijanima	COS	1 REAL	REAL
		DCOS	DOUBLE	DOUBLE
		CCOS	COMPLEX	COMPLEX
Tangens	tan(x) x u radijanima	TAN	1 REAL	REAL
		DTAN	DOUBLE	DOUBLE
Hiperbolni sinus	$(e^x - e^{-x})/2$	SINH	1 REAL	REAL
		DSINH	DOUBLE	DOUBLE
Hiperbolni cosinus	$(e^x + e^{-x})/2$	COSH	1 REAL	REAL
		DCOSH	DOUBLE	DOUBLE
Hiperbolni tangens	sinh/cosh	TANH	1 REAL	REAL
		DTANH	DOUBLE	DOUBLE

OPIS FUNKCIJE	MATEMATICKI OBLIK	IME FUNKC.	ARGUMENT (BROJ i TIP)		FUNKCIJA TIP
Apsolutna vrijednost	|x|	IABS	1	INTEGER	INTEGER
		ABS		REAL	REAL
		DABS		DOUBLE	DOUBLE
		CABS		COMPLEX	COMPLEX
Maksimalna vrijednost	max(x1,x2,...)	MAX0	>=2	INTEGER	INTEGER
		AMAX1		REAL	REAL
		DMAX1		DOUBLE	DOUBLE
		AMAX0		INTEGER	REAL
		MAX1		REAL	INTEGER
Minimalna vrijednost	min(x1,x2,...)	MIN0	>=2	INTEGER	INTEGER
		AMIN1		REAL	REAL
		DMIN1		DOUBLE	DOUBLE
		AMIN0		INTEGER	REAL
		MIN1		REAL	INTEGER
Ostatak dijeljenja x1 sa x2	x1(mod x2)	MOD	2	INTEGER	INTEGER
		AMOD		REAL	REAL
		DMOD		DOUBLE	DOUBLE
Prevodjenje u cjeli broj	INT(3.2)=3	INT	1	REAL	INTEGER
		IFIX		REAL	INTEGER
		IDINT		DOUBLE	INTEGER
		ICHAR		CHARACTER	INTEGER
Prevodjenje u realni broj	REAL(2)=2.	REAL	1	INTEGER	REAL
		FLOAT		INTEGER	REAL
		SNGL		DOUBLE	REAL
Prevodjenje u broj s dvostrukom preciznoscu		DBLE	1	INTEGER	DOUBLE
		DBLE		REAL	DOUBLE
Prevodjenje u kompleksni broj		CMPLX	1	INTEGER	COMPLEX
		CMPLX	ili	REAL	COMPLEX
		CMPLX	2	DOUBLE	COMPLEX
Prevodjenje u karakter	CHAR(7)='7'	ICHAR	1	INTEGER	CHARACTER
Zaokruzivanje	AINT(-5.8)=-5	AINT	1	REAL	REAL
		IDINT		DOUBLE	DOUBLE
Blizi cijeli broj	ANINT(3.4)=3.	ANINT	1	REAL	REAL
		IDNINT		DOUBLE	DOUBLE
Blizi integer	NINT(2.7)=3	NINT	1	REAL	INTEGER
		IIDNINT		DOUBLE	
Duzina CHARACTER varijable	za A='N.SAD' ILEN(X)=5	ILEN	1	CHARACTER	CHARACTER

Dodatak D.

Ogranicenja na poredak naredbi u programskoj jedinici



Iz ovoggrafickog prikaza je vidljivo mjesto naredbe u odnosu na druge. Komentar moze biti bilo gdje prije kraja programa. FORMAT naredba mora biti prije END a poslije naredbi: PROGRAM, FUNCTION, SUBROUTINE ili BLOCK DATA a bilo gdje izmedju naredbi desno.

I N D E X

Alfanumericki opisivac 62
Aritmeticka naredba grananja 39
Aritmeticke naredbe 29
Aritmeticki izrazi 15
ASCII karakteri 102
BACKSPACE naredba 74
BLOCK DATA potprogram 97
Bezuvjetni GO TO 36
Blok naredba grananja 41
Citanje i pisanje matrica 80
CLOSE naredba 76
COMMON naredba 85
CONTINUE naredba 36
DATA naredba 50
Datoteka 72
Dijagram toka programa 24
DO naredba 46
Dodijeljivanje pocetnih vrijednosti 49
DOUBLE PRECISION opisivac s pomicnim zarezom .. 61
Drugi oblik INTEGER opisivaca 59
Elementarne konstrukcije 6
END naredba 33
ENDFILE naredba 75
ENTRY naredba 96
EQUIVALENCE naredba 83
FORMAT naredba 56
Fortran naredbe 103
Funkcijski potprogram 92
Hijerarhija operatora 21
Hollerith opisivac 62
Implicitno odredjivanje tipa varijable 11
INTEGER opisivac 58
Ispis predznaka plus ("+") 61
Istrazivanje datoteka 80
Izracunati GO TO 37
Izrazi 15
Izvrsenje FORTRAN programa na terminalu 27
Konstante 7
Kontrola stampe 64
Kontrolne naredbe 33
Literal izrazi 17
Literal konstante 9
Literal naredbe 31
Literal opisivac 63
Literalne varijable 11
Logicka naredba grananja 39
Logicke konstante 9
Logicke naredbe 32
Logicke varijable 11
Logicki i relacijski izrazi 18
Logicki opisivac 61
Matrice 12
Nacin pisanja programa 23
Naredba READ za direktni pristup 77
Naredba READ za unutrasnje citanje 78
Naredba WRITE za direktni pristup 78
Naredba WRITE za unutrasnje citanje 79
Naredba za definiranje funkcija 90
Naredbe 28
Naredbe za bezuvjetni prelazak 36
Naredbe za dodijeljivanje vrijednosti 29
Naredbe za uvjetni prelazak 39
Numericke konstante 7

Numericke varijable 10
Opcenito 52
OPEN naredba 76
Opisivac za skok 63
Opisivaci polja 37
PARAMETAR naredba 86
PAUSE naredba 33
Ponavljanje grupe opisivaca 65
Potprogrami 87
Pozicioniranje na mjesto u slogu 63
Poziv potprograma 71
Prelaz na novi slog 64
Program 22
PROGRAM naredba 90
READ naredba 54
REAL opisivac s fiksnim zarezom 59
REAL opisivac s pokretnim zarezom 60
RETURN naredba 93
REWIND naredba 75
SAVE naredba 97
Simboli 6
Specifikacione naredbe 83
Specijalni REAL opisivac s pomicnim zarezom ... 61
Standardne fortranske funkcije 104
STOP naredba 35
SUBROUTINE naredba 94
Tretiranje praznih pozicija 61
Ulazno/Izlazne naredbe 52
Varijable 9
WRITE naredba 56
Zadaci 68

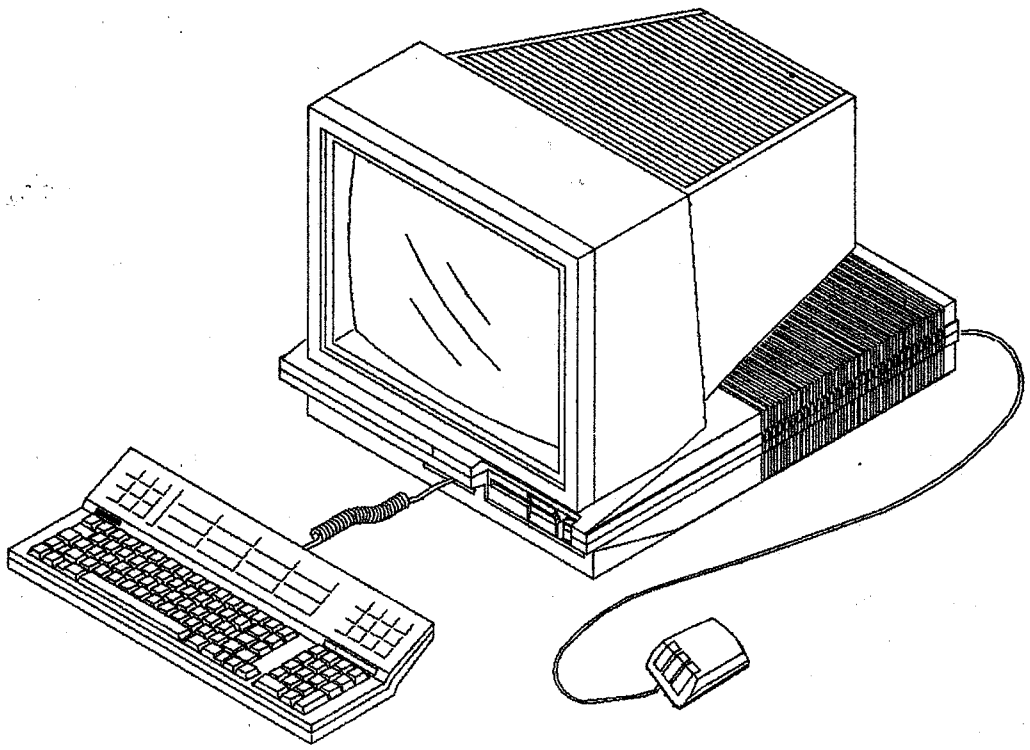
Prilog: Slike u knjizi

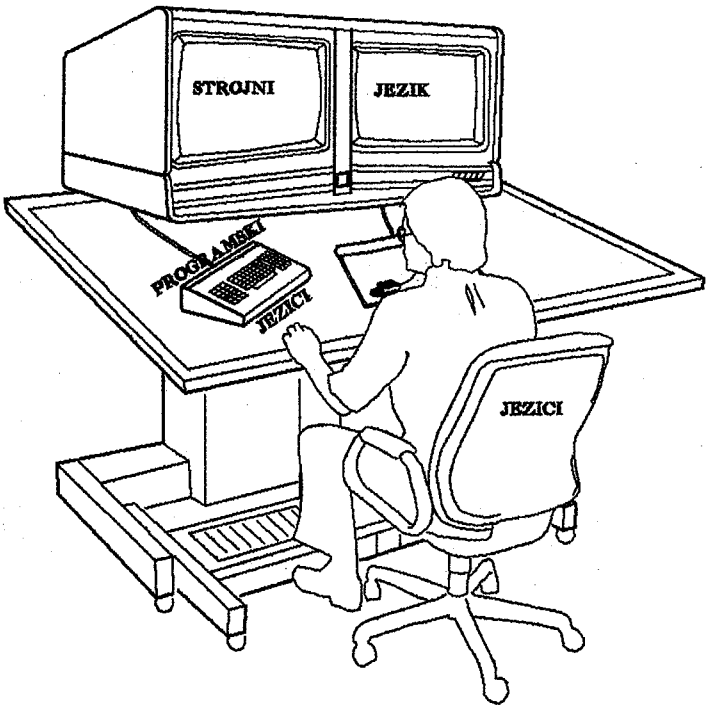
MILE PAVLIC

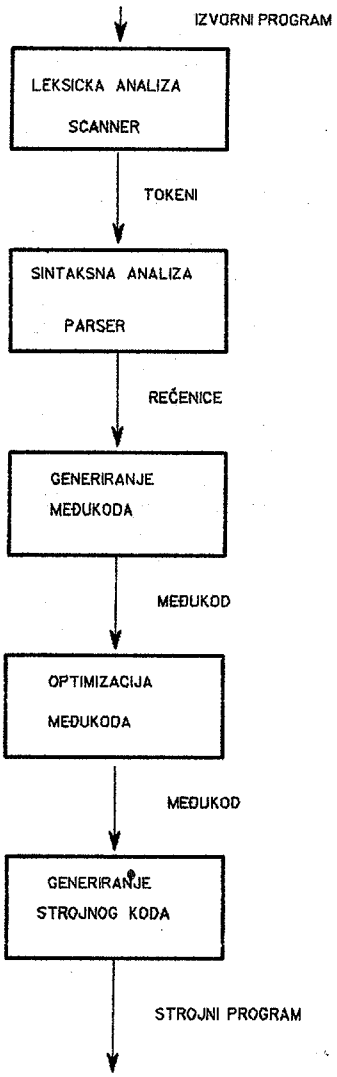
UVOD U

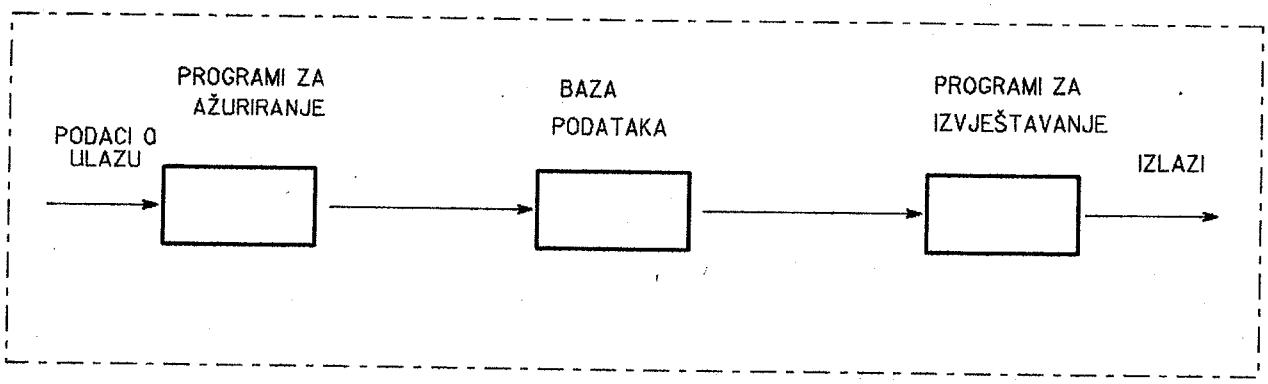
FORTRAN 77

ZA VELIKA I PC RACUNALA

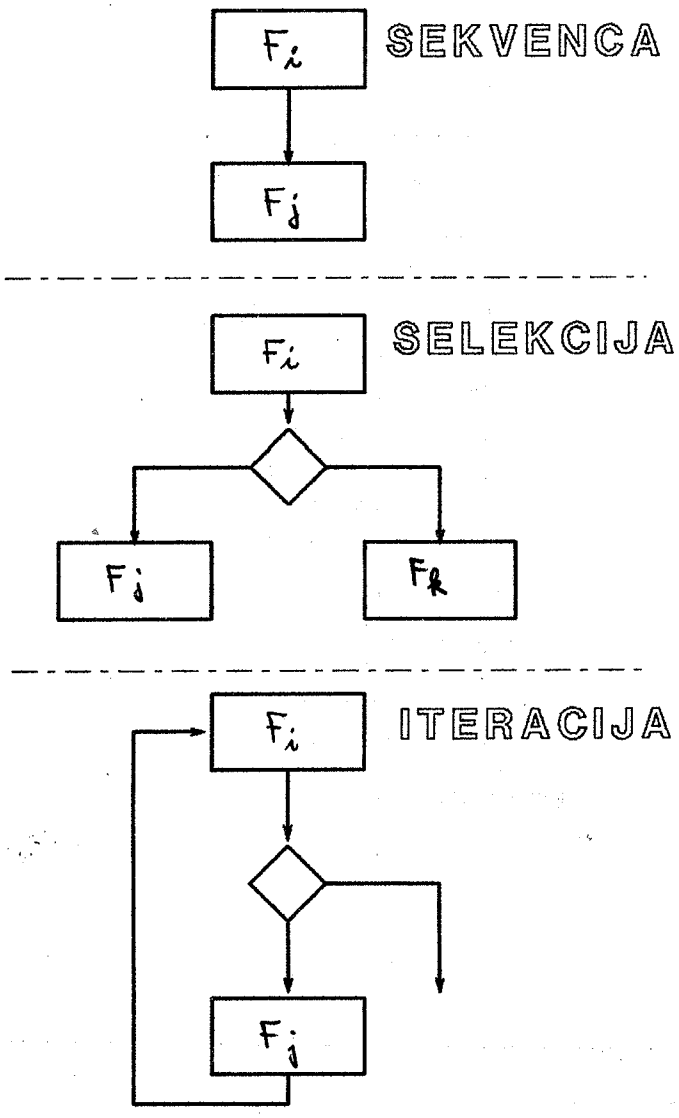


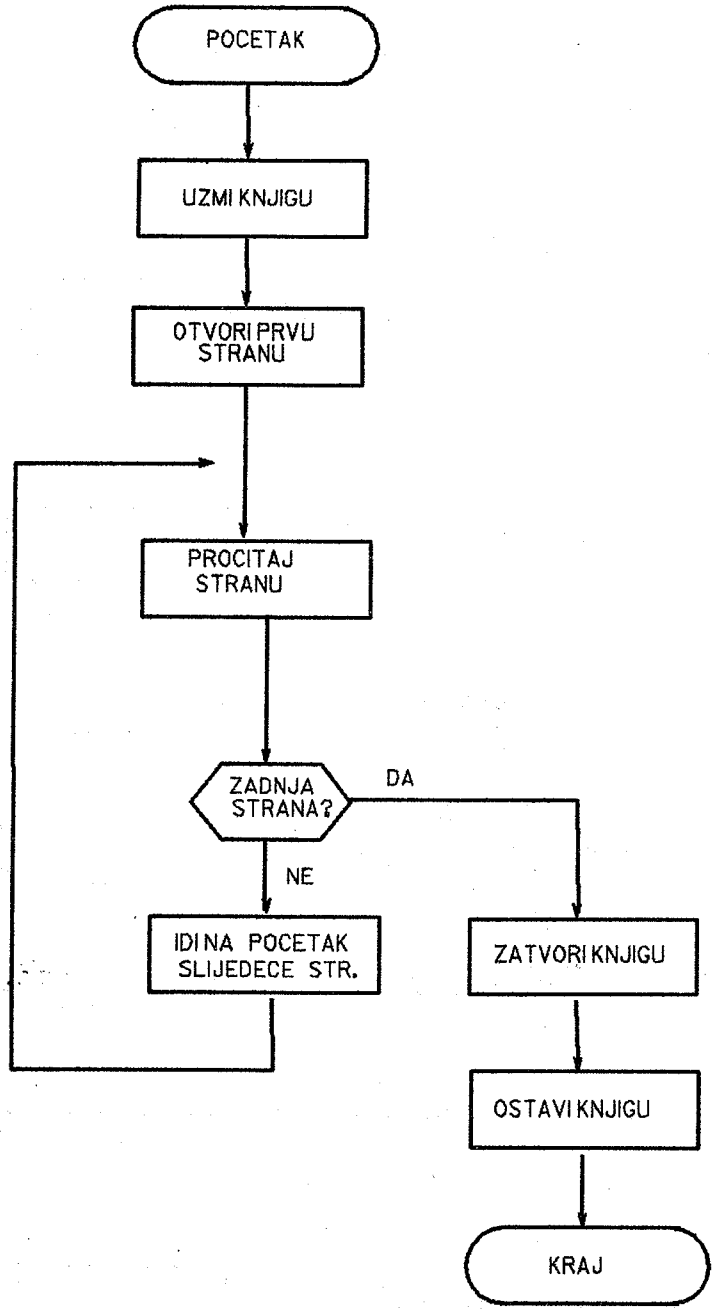

STROJNI
JEZIK
PROGRAMSKI
JEZICI
JEZICI


IZVORNI PROGRAM
LEKSICKA ANALIZA
SCANNER
TOKENI
SINTAKSNA ANALIZA
PARSER
REČENICE
GENERIRANJE
MEĐUKODA
MEĐUKOD
OPTIMIZACIJA
MEĐUKODA
MEĐUKOD
GENERIRANJE
STROJNOG KODA
STROJNI PROGRAM


PROGRAMI ZA
AŽURIRANJE
PODACI O
ULAZU
BAZA
PODATAKA
PROGRAMI ZA
IZVJEŠTAVANJE
IZLAZI

3.1


F_i
SEKVENCA
F_j
F_i
SELEKCIJA
F_j
F_k
F_i
ITERACIJA
F_j


POCETAK
UZMI KNJIGU
OTVORI PRVU STRANU
PROCITAJ STRANU
ZADNJA STRANA?
DA
NE
IDI NA POCETAK SLIJEDECE STR.
ZATVORI KNJIGU
OSTAVI KNJIGU
KRAJ

UNOS PODATAKA

ULAZ/IZLAZ PODATAKA

OPERACIJA (OPĆENITO)

ILI

ODLUKA

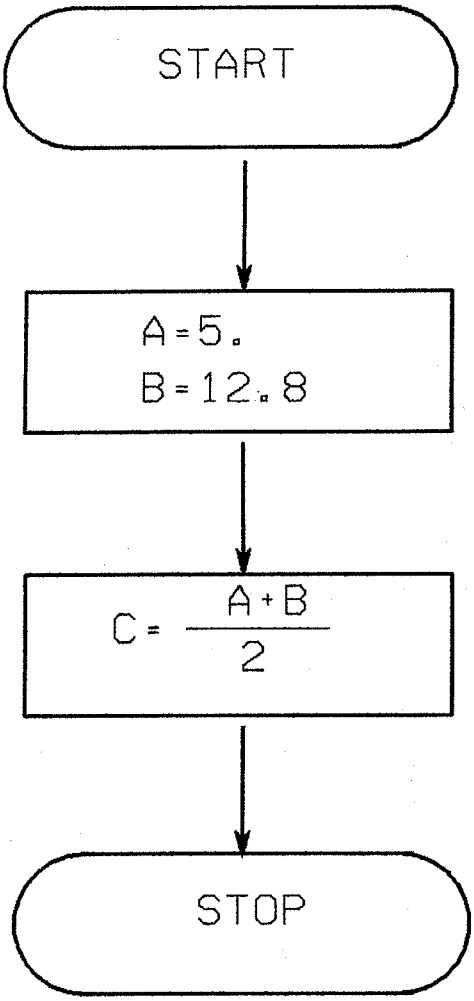
POČETAK

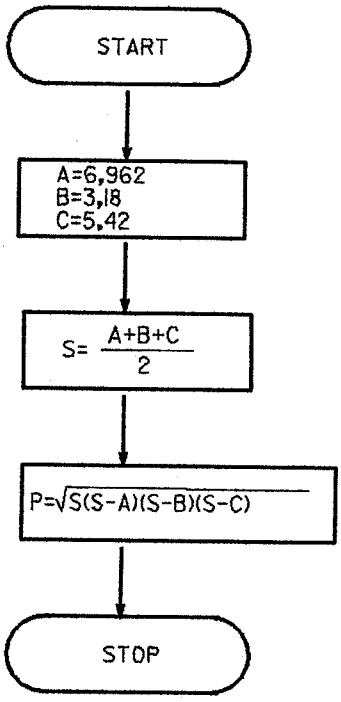
KRAJ PROGRAMA

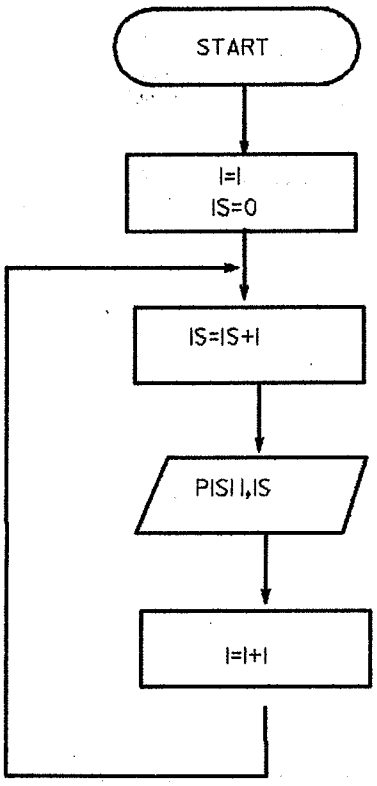
PRIKLJUČNA TOCKA

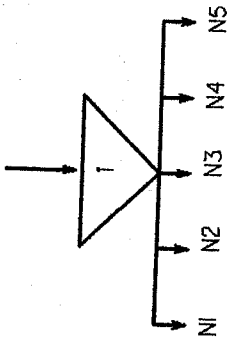
LINIJA ODVIJANJA OPERACIJA

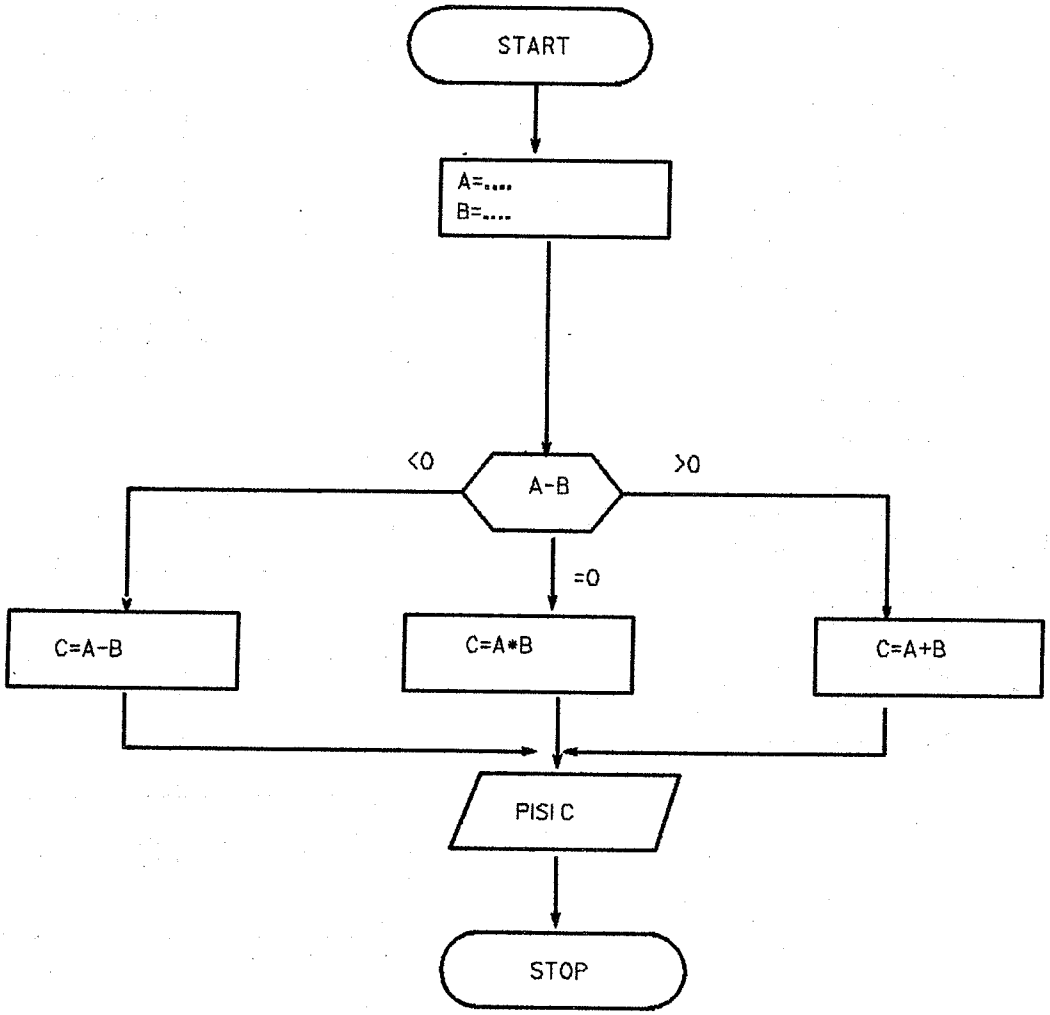
POTPROGRAM

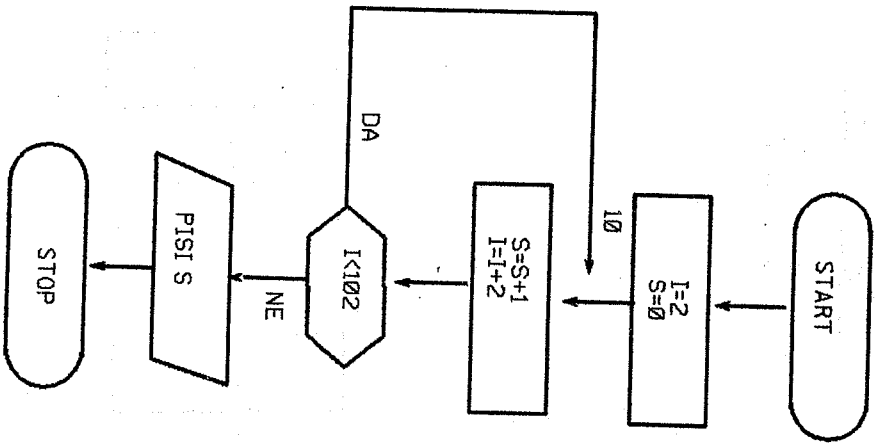

START
A=5.
B=12.8
C= A+B / 2
STOP

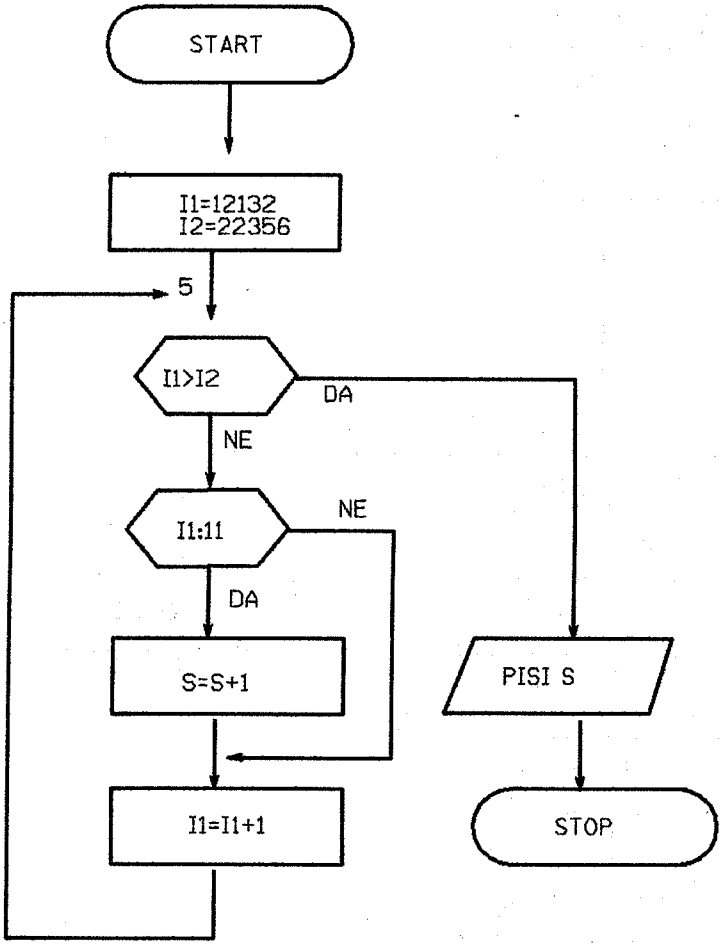

START
A=6,962
B=3,18
C=5,42
S= A+B+C / 2
P=√S(S-A)(S-B)(S-C)
STOP

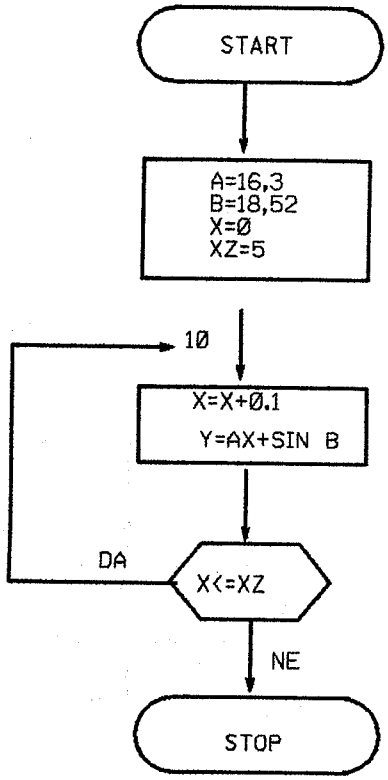

START
I=1
IS=0
IS=IS+I
PISI I,IS
I=I+1

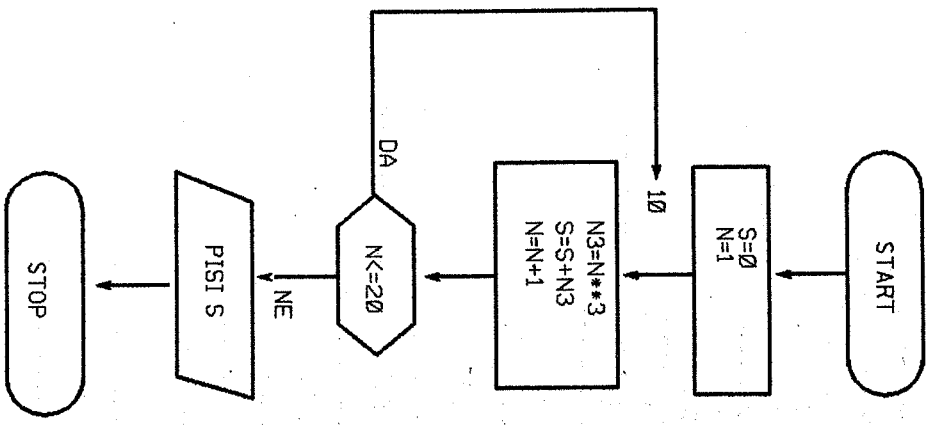

I
N5
N4
N3
N2
NI

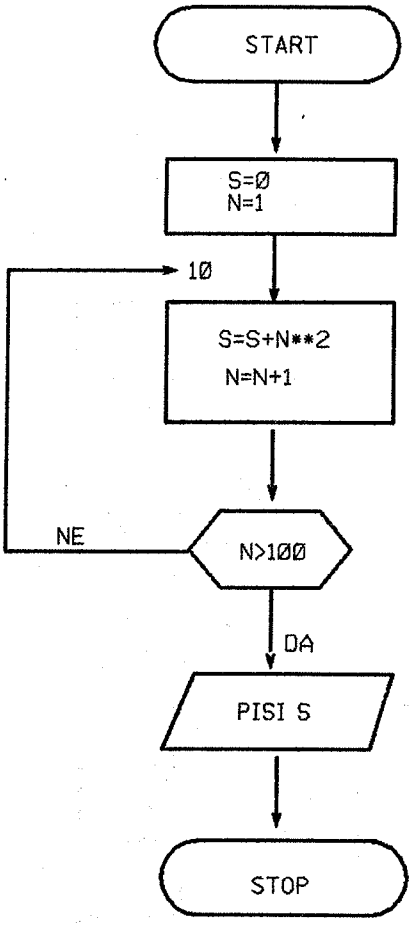

START
A=....
B=....
A-B
<0
=0
>0
C=A-B
C=A*B
C=A+B
PISI C
STOP

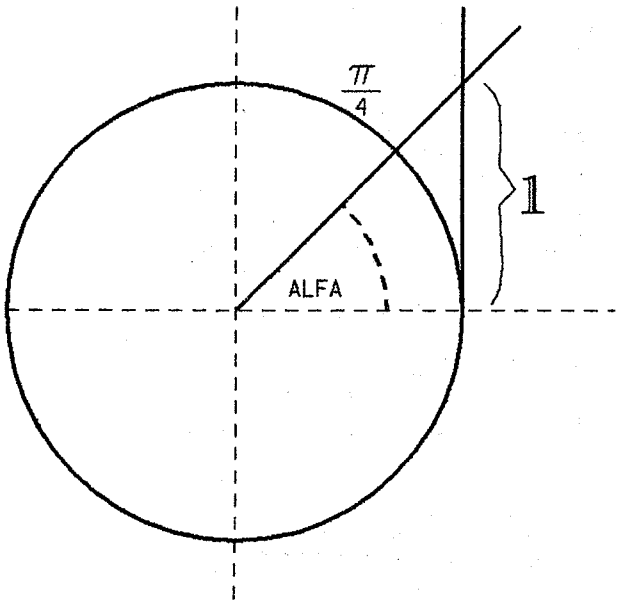

START
I=2
S=0
S=S+1
I=I+2
I<102
DA
NE
10
PISI S
STOP

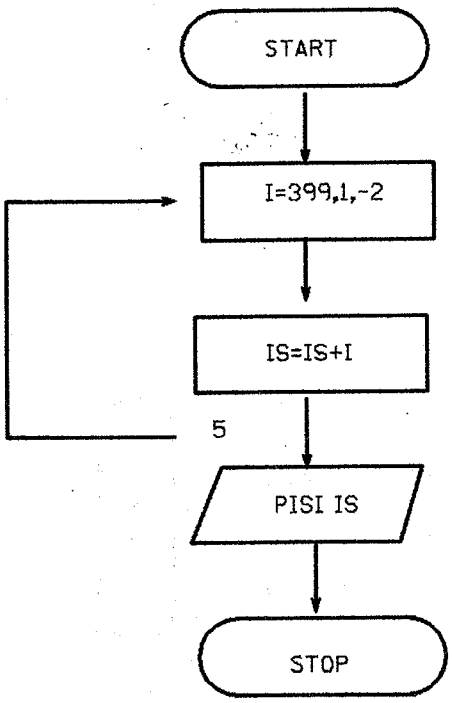

START
I1=12132
I2=22356
5
I1>I2
DA
NE
I1:11
NE
DA
S=S+1
I1=I1+1
PISI S
STOP

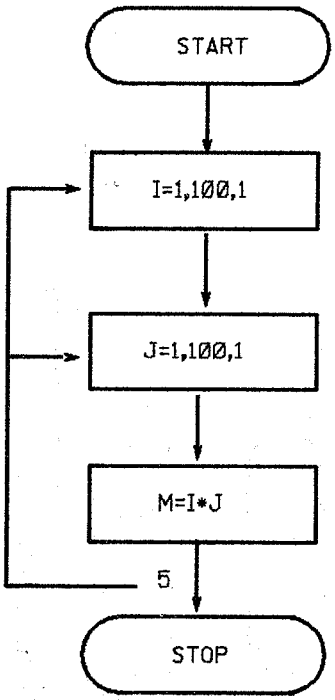

START
A=16,3
B=18,52
X=Ø
XZ=5
1Ø
X=X+Ø.1
Y=AX+SIN B
DA
X<=XZ
NE
STOP

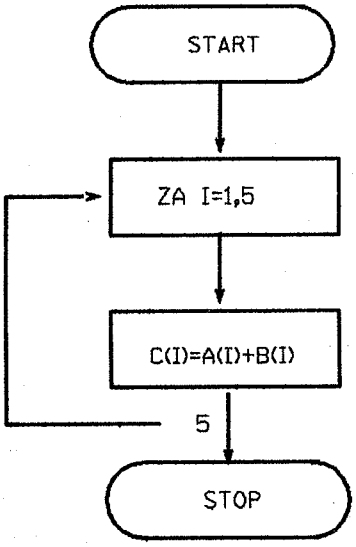

START
S=Ø
N=1
1Ø
N3=N**3
S=S+N3
N=N+1
N<=2Ø
DA
NE
PISI S
STOP

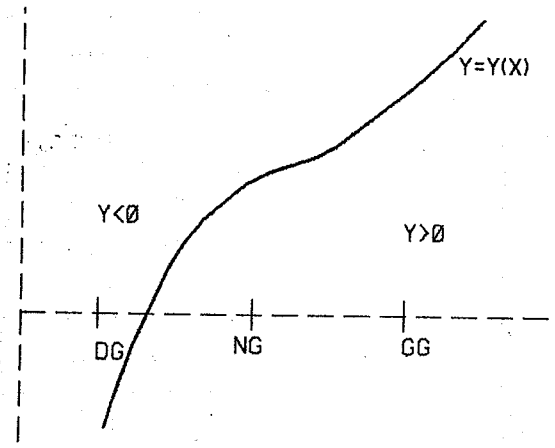

START
S=Ø
N=1
1Ø
S=S+N**2
N=N+1
NE
N>1ØØ
DA
PISI S
STOP

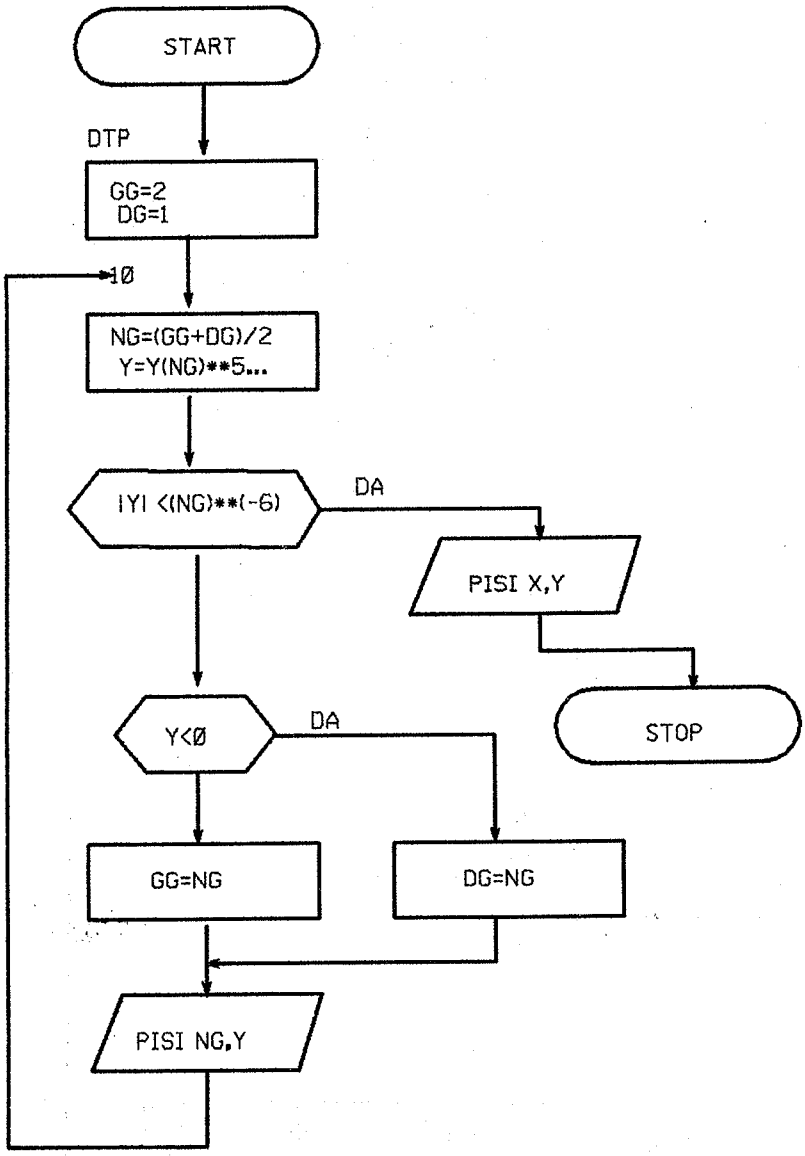

π/4
1
ALFA

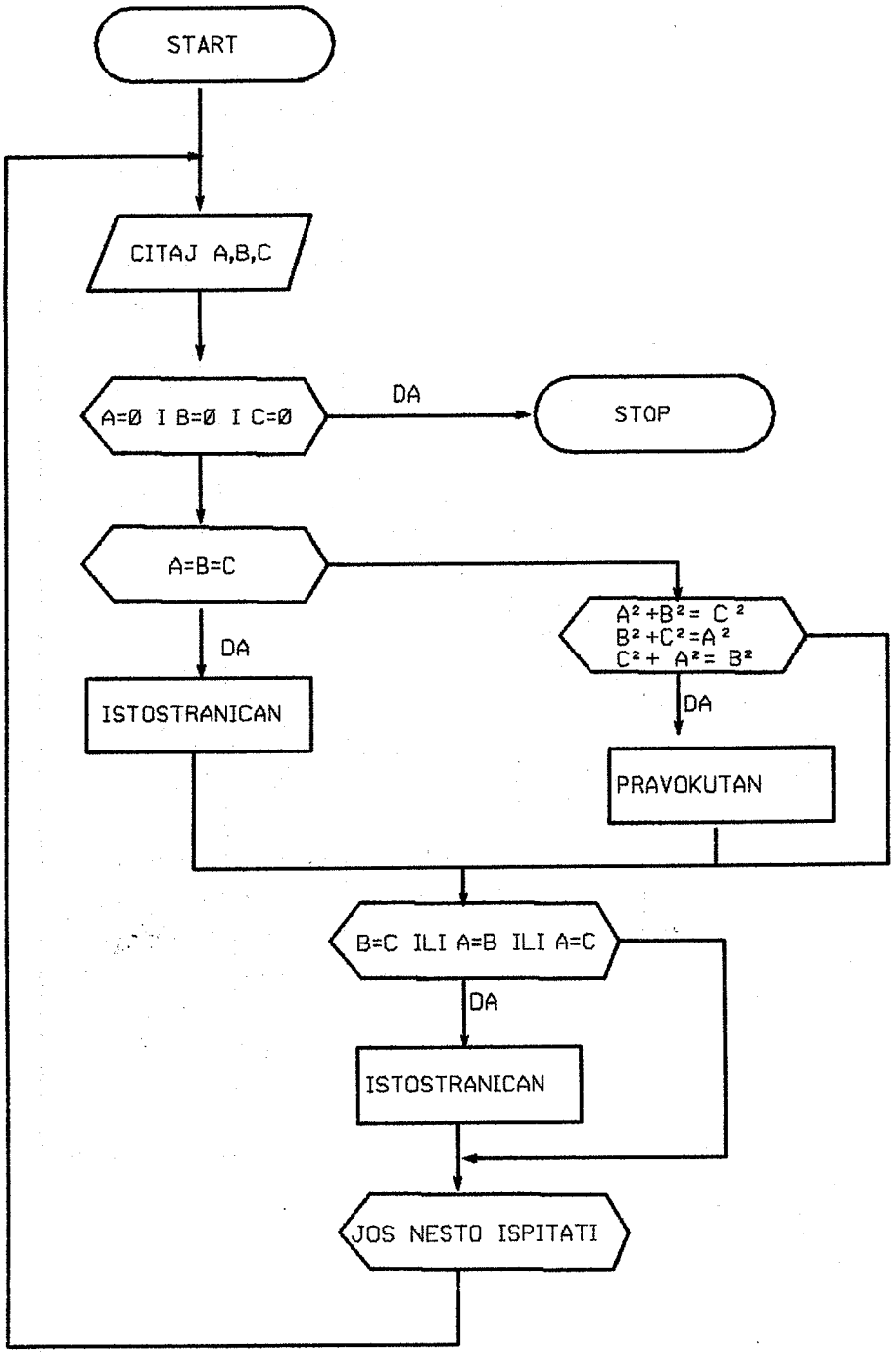

START
I=399,1,-2
IS=IS+I
5
PISI IS
STOP

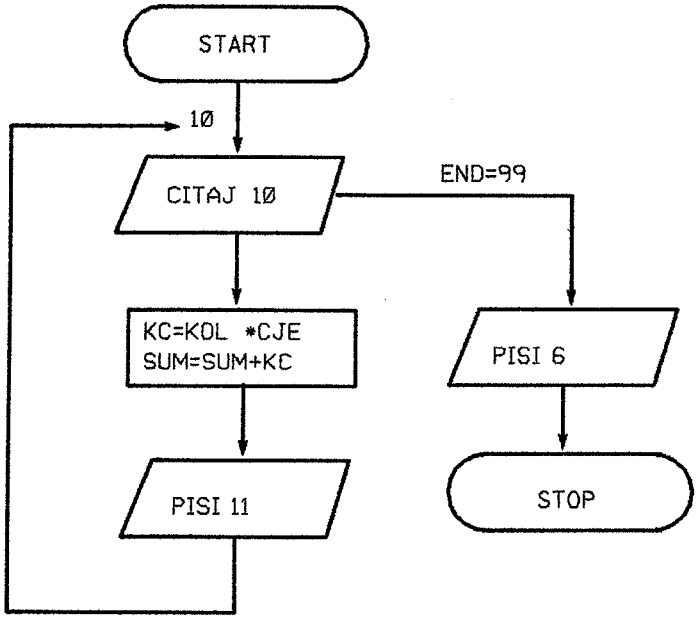

START
I=1,1ØØ,1
J=1,1ØØ,1
M=I*J
5
STOP


START
ZA I=1,5
C(I)=A(I)+B(I)
5
STOP


Y=Y(X)
Y<Ø
Y>Ø
DG
NG
GG


START
DTP
GG=2
DG=1
10
NG=(GG+DG)/2
Y=Y(NG)**5...
|Y| <(NG)**(-6)
DA
PISI X,Y
STOP
Y<0
DA
GG=NG
DG=NG
PISI NG,Y


START
CITAJ A,B,C
A=Ø I B=Ø I C=Ø
DA
STOP
A=B=C
DA
ISTOSTRANICAN
A²+B²= C²
B²+C²=A²
C²+ A²= B²
DA
PRAVOKUTAN
B=C ILI A=B ILI A=C
DA
ISTOSTRANICAN
JOS NESTO ISPITATI



GLAVNI PROGRAM

```
.....
.....
.....
.....
.....
END
```

BLOCK DATA POTPROGRAM

```
BLOCK DATA
.....
.....
.....
END
```

FUNKCIJSKI POTPROGRAM

```
....FUNCTION
.....
.....
.....
END
```

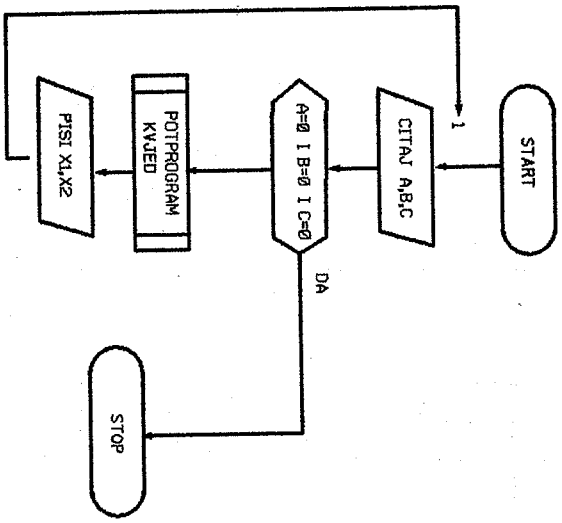
POTPROGRAM

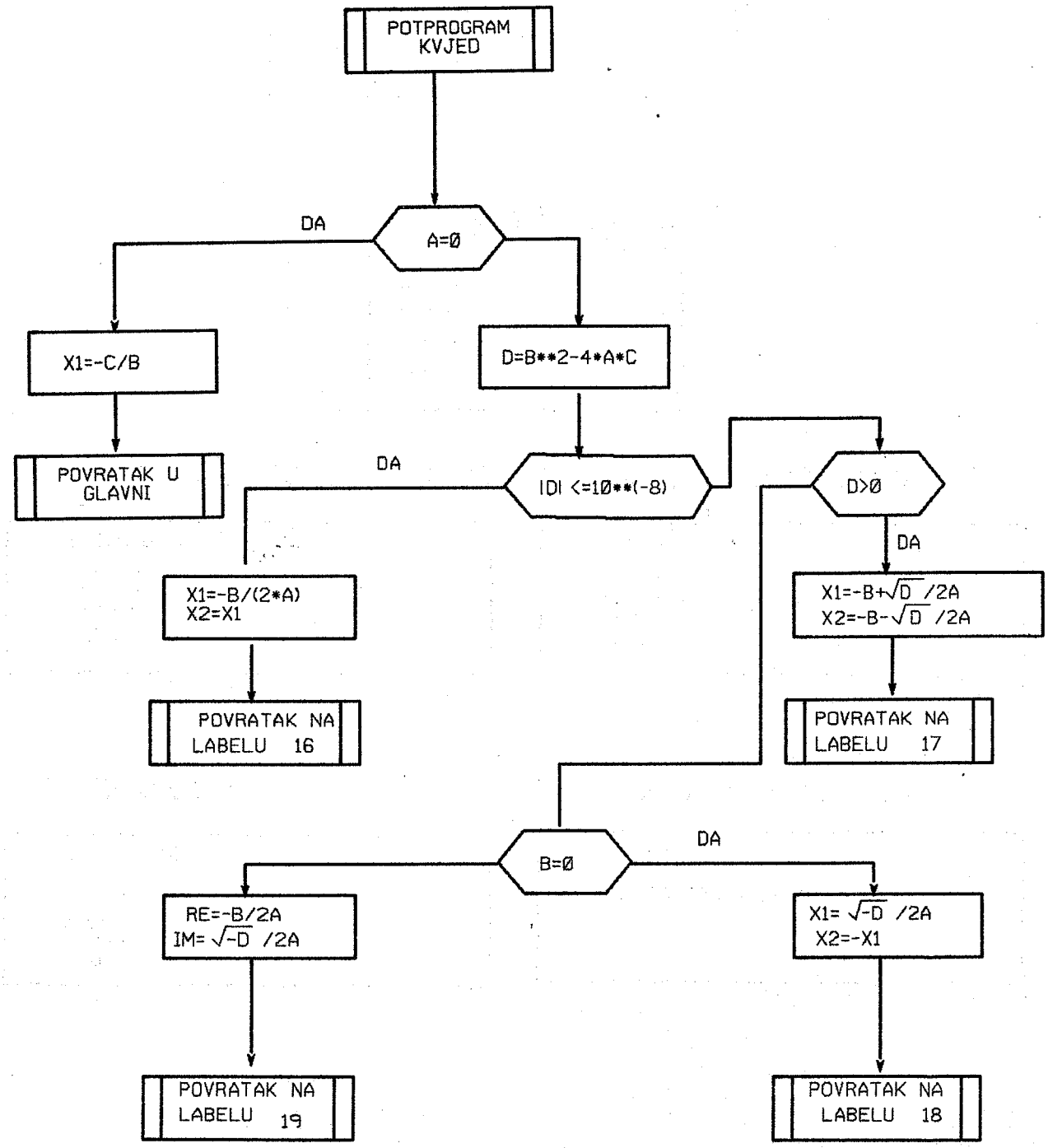
```
SUBROUTINE
.....
.....
.....
END
```

VANJSKE PROCEDURE

PISANE U

NE-FORTRAN JEZIKU


START
CITAJ A,B,C
A=0 I B=0 I C=0
POTPROGRAM KVJED
PISI X1,X2
DA
STOP


POTPROGRAM KVJED
A=Ø
DA
X1=-C/B
POVRATAK U GLAVNI
D=B**2-4*A*C
IDI <=1Ø**(-8)
DA
X1=-B/(2*A)
X2=X1
POVRATAK NA LABELU 16
D>Ø
DA
X1=-B+√D /2A
X2=-B-√D /2A
POVRATAK NA LABELU 17
B=Ø
DA
RE=-B/2A
IM=√-D /2A
POVRATAK NA LABELU 19
X1= √-D /2A
X2=-X1
POVRATAK NA LABELU 18

L I T E R A T U R A :

1. Sperry Rand Corporation,"FUNDAMENTALS OF FORTRAN Programmer Reference", USA 1970.
2. Sperry Rand Corporation,"FORTRAN V Programmer Reference", USA 1973.
3. Sperry Rand Corporation,"FORTRAN V Library Programmer Reference", USA 1974.
4. Sperry Rand Corporation,"FORTRAN(ASCII)Programmer Reference", USA 1978.
5. Sperry Rand Corporation,"FORTRAN (ASCII)Level 10.R1 Programmer Reference", USA 1982.
6. Sperry Rand Corporation,"FORTRAN (ASCII)Level 11R1 Programmer Reference", USA 1985.
7. Dr ing. Bozidar Stefanini, "FORTRAN udžbenik programiranja",Zagreb 1970, Tehnička knjiga.
8. Dr Nedeljko Parezanović, "Računarske mašine i programiranje programski jezik FORTRAN IV", Beograd 1982, PFV.
9. Veljko Vuletić, Čamila Ljubović, "PROGRAMIRANJE /FORTRAN/", Sarajevo 1984, Svjetlost.
10. Davorin Fulanović, "Uvod u fortran", Zagreb 1985, Prosvjeta.
11. Zavod za primjenu elektroničkih računala, "Praktični primjeri programiranja (fortran)", Zagreb 1981/82, Centar za stručno usavršavanje kadrova za automatsku obradu podataka.
12. 12.. Microsoft Corporation ,"Microsoft FORTRAN Reference Manual" , 1985.
13. Microsoft Corporation, "Microsoft FORTRAN Compiler for MS-DOSth Operating System User's Guide", 1985.
14. Roberto Doretti, Roberto Farabone, "Dal FORTRAN IV al F0RTRAN77", Milano 1983, Edizione Italiana.
15. S.Lipschutz,A. Poe,"Programmare in FORTRAN", Milano 1985,ETAS Libri.
16. Michel Dreyfus, "FORTRAN IV", Milano 1985, Edizione Italiana.
17. P. Ridolfi, " Applicazioni del Fortran ", Milano 1982,Ffanko Angeli Editore.
18. Stanoje Bingulac, Marko Vusković, "Programski jezici i prevodioci", Beograd, 1985, istitut "Mihajlo Pupin".
19. Vidojko Čirić,"Uvod u programiranje i programski jezik FORTRAN",Beograd,1986,Naučna Knjiga.
20. Branislav Lazarević i dr "Projektovanje informacijskih sistema", Beograd 1985, Naučna Knjiga.
21. Branislav Lazarević, "Uvod u informacione sisteme", Beograd 1982, FON Univerzitet u Beogradu.
22. Pavlić, M., Zemljić, V., Srdoč, A., Ledinko, Z., & Vučić, B. Informacijski sustav praćenja pristizanja i ugradnje opreme platforme LABIN, 1983.
23. Mile Pavlić, Metode eksponencijalnih poravnavanja. In 5. međunarodnog simpozija" Kompjuter na sveučilištu". Hrvatska znanstvena bibliografija i MZOS-Svibor, 1985.
24. Vinko Zemljić, Mile Pavlić, Alira Srdoč Primjena mrežnog planiranja u gradnji platforme Labin. 1985.
25. Mile Pavlić, Zlatko Jugo, Primjena modela eksponencijalnih poravnavanja za predviđanje vremenskih serija. VII simpozij teorija i praksa brodogradnje in memoriam prof. Leopold Sorta, 1986.
26. Mile Pavlić, Vladan Jovanović, Predrag Dizdarević, Zlatko Jugo, Projektiranje IS kalkulacija cijene broda primjenom metoda SSA i MOV. III savjetovanje o informatičkoj djelatnosti, 1986.